UPDATE!

$4.95 July 1989

Subscription $18.00

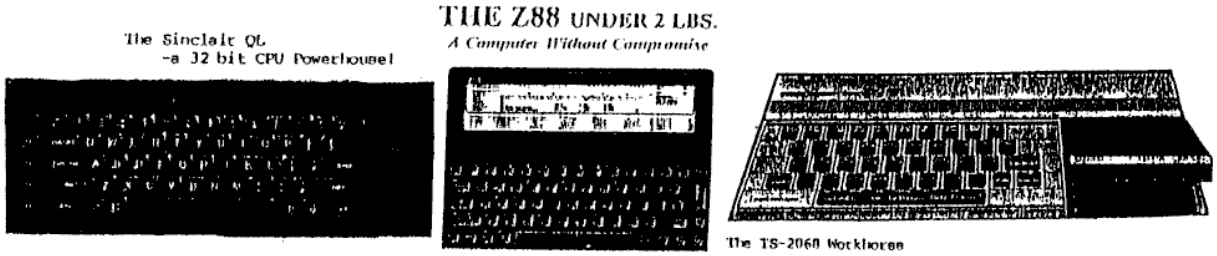
ComputerSystems

Take it to the Limit

A Quarterly Magazine for the users

OF

SINCLAIR COMPUTERS



Features this Issue

ANNOUNCEMENTS: 3, 4, 11

For the Sinclair QL

Abacus Check Balancer: 36
Amateur QSO File: 43
Auto Loader: 16
Cable Column: 17
(Filer Program)
List Scan: 39
LLIST: 42
Perks and Quirks: 22
Power Fix: 9
Printer Set-Up: 37
QL Media: 42
3 New QL ISSUE DISKS: 48

For the Cambridge Z88

Memory Map organization: 26
News Notes: 25
Perks and Quirks: 24
Z88 Battery change: 16

For the TS-2068

Daisy.B6 Software: 47
D.Base-1 Software: 27
FD-68 Update: 12
LKDOS Tools: 44
Plotting: 20
Universal ASCII FILE
and Code Printer: 10
View Calc for Disk
Drive: 32
4 New Issue Disks: 49

Update Magazine

1317 Stratford Ave.
Panama City, FL 32404
904 871 3556

NEWS AND NEW PRODUCTS

LKDOS SOFT-WARE

MAXCOM 300/1200 baud Terminal/BBS

This full feature modem software lets you effectivly link the modem to the disk, so you can upload or download files larger than 100K.This lets you send NMI saves and large text files as well as Basic or Code files, which other modem programs couldn't send because of memory limitations. Also files can be sent with or without a information header (Spec-term compatible) .

It also has a 64 column display and is the only 2068 modem program that will run at full 1200 baud without missing characters. Other features of the terminal mode are IBM graphic characters, auto-dialer, macro keys, 110K disk buffer for captured text, auto repeat keys.

The BBS is the most elaborate available for the 2068 and allows the remote user full access to the disk drives similar to a CPM BBS. Features of the BBS are - 300/1200 baud, operation with fast reponce even at 1200 baud, Passwords, priority levels, Use time limits, Multiple Message Bases, Uploads and Downloads are not limited by Memory, Dual Window Monitor screen, Interupt driven time out and Hangup protection, Also Keeps a User log of callers Names and Activity on disk so a printer isnt needed and its easily customised and expanded.

MAXCOM can be used with the 2050 modem or with the Z-SIO RS232 interface and a 300/1200 baud modem. (a version for the Aerco RS232 is in the works).

PRICE: $24.95

LARKEN DISK EDITOR

This program lets you examine or change any byte on a LKdos disk. It is useful in repairing damaged or glitched disks, changing the disk Name or head speed etc. You can also examine the track map and disk info headers.

The documentation for the disk editor also includes information on how to access LKdos from Machine Code and information on the structure of the disk catalog and data blocks.

PRICE: $15.00

SEQUENTIAL/RANDOM Access FILES

This utility is a ram resident extension to the LKdos operating system and lets you create, read from, or write to very large data files kept on disk. It uses the commands OPEN#, CLOSE#, PRINT#, INPUT#, and INKEY$# to access the files from BASIC. There is also a high speed search command for finding a text string within the data file. Using the TAB command, you can randomly access directly any record in a file.

The utility is easily used within your basic programs and has a lot of uses, from storing large data files, creating large spell checker dictionarys or even converting you basic programs to an ascii text file .

This utility lets you take the limits off of the data capacity of the 2068 .Files on disk can be huge !

A simple data base program is included on the disk to demonstrate the utility.

PRICE: $ 15.00

Coming Soon from Larken Electronics

** The Larken Desk Top Publisher ** Produce high quality printing (like this page) with an epson compatible printer and LKdos. The entire Hi-res graphic page is kept on the disk and scrolling though the page is as easy as clicking the joystick or mouse on the icon type menu. Text and Graphic editing with multiple sized fonts will also be featured.

** Spell Checker for Tasword and M-Script ** This utility program will let you have your wordprocessor text files be checked by a 350K dictionary.

All Prices $US - Specify type of drive when ordering.

LARKEN ELECTRONICS RR#2 NAVAN, ONTARIO CANADA, K4B-1H9

(613)-835-2680

QUANTUM LEVELS MAGAZINE

It was announced at the recent CATS FEST that Quantum Levels magazine would publish two more issues and cease publication. I have been a subscriber to QL's since its first issue and I regret that it is ending. Quantum Levels began publishing before Sinclair sold the QL to Amstrad, who then stopped production. This came at a time when QL Computing was just getting started good in North America. As QL production ended so did the prospects for success of a publication devoted exclusively to the support of QL users. That is the same "domino effect" that Timex Support experienced when the TS-2068 production was suddenly stopped. We supporters hang on as long as the dwindling numbers allow, and then must fold the cards. We had hoped that the combining of Syncware News and Quantum Levels would see them through.

Hang ON to our Resources!

Tom Bent, the editor of Quantum Levels, is the leading QL technical expert in North America. Tom is also the President of the Capitol Area User Group (CATS), and is the Quanta Librarian for North America. Tom has a deep commitment to QL Computing and support. It is in the best interests of all QL users that we keep Tom Bent's expertise available to us for as long as the QL user Group can maintain its enthusiasm. With that objective in mind I have written to Tom Bent inviting him to participate with Update Magazine so that we will keep his expertise and services readily at hand. Drop Tom a line if you will and urge him to participate with us. (Make him feel wanted!) Also, several have ask me for Tom's address to send their QL's for ROM upgrade. TOM BENT, 9016 Flicker Place, Columbia, MD 21045. Hang in there with us Tom!

NEW ISSUE DISKS - QL and TS-2068

NEW Update issue disks are unveiled in this July 89 issue for the TS-2068 users and for the QL users. TS-2068 issue disks are available in three disk formats, LKDOS, Oliger DOS, and Aerco FD-68, 40 or 80 track, on 5 1/4" disk.

QL issue disks are available in 80 Track, 5 1/4" Floppy or in Micro drive Cartridge. If ordering for micro drive, please inclose a formatted mdv cartridge for dubbing.

The prices for all issue disk ensembles is $22.00 (includes shipping). Satisfaction is guaranteed or money back. See the index for full discription of the issue disk programs. Also check the index of past Update issues for discription of other issue disks in the Update Library. The library has a good variety and has grown to a size that it is no longer feasable to describe all issue disk ensembles in each magazine issue.

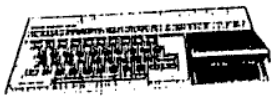
The Issue Disk program is a joint effort by Update Magazine and individual programmers to provide QUALITY NEW programs and utilities for the users. Pricing is to cover expenses and to provide $11.00 per disk to the programmer of the principal program included in the disk. This is Update Magazine's version of the ShareWare concept. The Issue Disk concept was designed to encourage programmers to PRODUCE, while taking the burdens of advertising, literature production and mailing from their shoulders. Each program and utility is documented in the pages of Update in the form of a "mini manual". Shorter programs may be key-in projects with detailed program analysis. The programs are tested at Update to insure proper operation.

The Issue disk programs are not in public domain and they remain the property of the author unless specifically given to public domain within the text of the documentation. Some authors use the issue disk as their first outlet for the software and follow on with separate marketing efforts. All programs are original, are NEW and are well tested. When you buy an Issue Disk you get more than your money's worth of useful software, and you help to encourage a programmer, to continue to support us all. If no author's name appears in the documentation of a program it signifies a program authored by Update Magazine in-house. Some are produced jointly by Update and another programmer. An example of this is the "DBase 1" software in this issue, produced by Update and assisted by Don Lamen. Don will share in the proceeds. The Issue Disk program is unique. You get excellent new software, and our programmers are encouraged to PRODUCE.

SUBSCRIPTION RENEWAL TIME

This is the last issue of the 1989 issue year, and is your last issue unless you have already renewed your subscription. Please RENEW NOW. Update needs the money to pay for the next issue publication. Subscription renewal is $18.00 for a year of issues (October through July).

With your renewal please state your Computer type and preferred Disk Drive format (40 or 80 tracks). We attempt to keep a data base with that information for your support.



"Mlz.B4" (A Mlzzery Saver)

OR- A program for Programmers

Dontated to Public Domain by:

Update Magazine

This one is for the hard working programmers who contribute their works to magazines or to their user group. We will set the scenario. You have worked away with an assembly program and finally it is checked out and working. You are tired of the program, and you'd like to go on to something else. But you aren't through yet! It is pure misery to have to construct a LOADER program with a table of Decimal Code numbers to POKE the code to memory and save the code program to disk. But you have to do that in order to tell people how to work your code into memory and into a disk file.

We will name this program "Mlz.B4", because it relieves you of creating a loader program "that B-4 Now has caused you so much Mlzzery"- (mlz.B4 now, that is).

"Mlz.B4" publishes your code loader/saver in the form of a Basic LLIST of the loader program. The data lines are created by PEEKING your Code program from memory. Your "pupil" who types the program in is given the Loader already prepared to key in. Mlz.B4 provides a check sum at the end for the "Joe or Jane" to key in to check their key-in accuracy. And YOU, the programmer, can go on to working on another program without the mlzzery of having to create that long table of decimal numbers. For a proof of this, look at the "LOADER PROGRAM LISTING" given with the program "TS-2068 Data Input" elsewhere in this issue. That Loader program was created by "this-un".

As most programmers can tell, line 90 initializes LKDOS for printing a 32 column LLIST, and lines 100 and 250 uses LKDOS syntax for LOAD and SAVE. Change these lines to suit your DOS requirements.

Instead of a lot of other line number explanations, just key in the program and work it on one of your code programs that you know the starting address number in memory and the Byte Length. Then <RUN> and watch it create your LOADER/SAVER LLIST Prgram, complete with Filled-in DATA lines, ready to publish. Have some fun next time, instead of working on another Loader program and decimal table after your next MC Code creation. _BJ-

```
 80 REM **** Create a Loader
program with DATA lines for a
CODE Program in disk. ****

 85 REM ** Donated to Public
Domain by Update Magazine **
"
 90 BORDER 0: PAPER 0: INK 7: C
LS : RANDOMIZE USR 100: OPEN #3,
"lp": RANDOMIZE USR 100: POKE 16
096,0: RANDOMIZE USR 100: POKE 1
6090,31: RANDOMIZE USR 100: POKE
 16092,0: RANDOMIZE USR 100: POK
E 16094,8: LPRINT
 100 LET a=2000: LET p=0: RANDOM
IZE USR 100: CAT ".C1",: INPUT "
Code File + EXT?";x$: RANDOMIZE
USR 100: LOAD x$CODE : INPUT "St
art Adr or Where to SAVE the Co
de?";x1: INPUT "Byte Length?";x2
: LET x=0
 105 LLIST 200
 110 FOR n=x1 TO x1+x2-1 STEP 20

 120 LPRINT a;" DATA ";: LET a=a
+2
 124 FOR y=n TO n+19: LET p=PEEK
 y: IF y>x1+x2-1 THEN  GO TO 170

 126 IF y<n+19 THEN  LPRINT p;",
";: LET x=x+p
 128 IF y=n+19 THEN  LPRINT p: L
ET x=x+p
 130 NEXT y
 160 NEXT n: STOP
 170 LPRINT p: LET x=x+p: LPRINT
 a;" REM Check sum=";x: STOP
 200 REM *** LOADER PROGRAM
           LISTING **
 210 CLS : INPUT "Input Check Su
m of Program";ck: INPUT "Input S
tart Addres to SAVE Code ";x1: I
NPUT "Input Byte Length:";x2
 220 RESTORE 2000: LET x=0: FOR
n=x1 TO x1+x2-1: READ a: LET x=x
+a:: POKE n,a: REM *Poker*
 225 NEXT n
 230 LET b=0: FOR n=x1 TO x1+x2-
1: LET b=b+PEEK n: REM * Get Act
ual Sum *
 232 NEXT n
 234 IF b=ck THEN  GO TO 250
 240 BEEP 2,6: PRINT "ERROR!  Re
check work.": STOP
 250 CLS : INPUT "Key in a Code
Name To SAVE.      Include Extens
ion-";a$: INPUT "Input Starting
Address for SAVE ";x: RANDOMIZE
USR 100: SAVE a$CODE x,x2
 260 CLS : PRINT AT 10,0;"The co
de program has been SAVED at add
ress ";x: STOP
```

See Page 50 for "Data Input" and its Loader created with this program.

Window Defining: A Program, part I

The Sinclair QL Advanced Programming

by David McCall

When programming, to get the full power out of your Sinclair QL one must learn how to effectively use its windows. Using the SuperBASIC commands can become tedious and slow. This series of four articles will develop a program to completely define any window giving all relevant facts about it. The first segment will give a general overview and discuss the assembly language routine that will be used. The second installment will be about the required functions. The third will discuss the procedures used and the fourth will be the main program. Each segment will be a brief tutorial on the relevant subject while at the same time presenting some useful features the can be used in your own programs.

The window defining program that will be developed started out as a simple assembly language routine to size and move a window on the screen but, other details about the window were continually being needed in the development of SuperBASIC programs. Details such as how many characters can be printed in the window with the desired character size and border width. With the program this calculation can easily be displayed with any of the parameters being changed and giving a new answer immediately. Other laborious problems when done strictly in SuperBASIC such as scaling the screen and positioning the origin, determining maximum and minimum point positions, etc., can be avoided after the next four articles and the program is completed. In the process of developing the utility, you will obtain a better understanding of SuperBASIC's structured programming potential and the integrated use of functions, procedures, and machine language calls.

The first thing needed to give the program a professional quality and save any relevant data on the screen as the window is being moved and defined is a window swapping routine. This needs to be done quickly and efficiently. Of course, assembly language will be the best choice. However, before we start to write the routine we need to know how the screen is stored in memory.

Screen memory starts at location 131072 and has 512 by 256 pixels stored in 32 continuous kilobytes. Eight pixels are stored per every word (16 bits or 2 bytes). Depending on the mode, it may seem as if there are only four pixels per word but, two pixels are combined to achieve more colors. The format of the word is not very important to the swapping routine because, it will swap only on eight pixel or word boundaries. This choice was made to make the routine work as fast as possible. More information about the format is given on page 30 in the Concepts section of the QL manual.

The routine will merely copy one noncontinuous region of memory to a continuous region. In our case, the noncontinuous region will be part of the QL's screen memory. It is noncontinuous because the window may not fill up the entire screen. It is being saved in a continuous fashion just like the entire screen for the QL is stored, thus saving memory and time when the copying takes place. With a few minor changes it can be used for other purposes. The basic algorithm for performing the swapping is: 1) Determine starting location of the window in memory; 2) Copy the first row of pixels from the screen's memory to the storage memory for the window, or vice versa; 3) Continue copying rows until the entire window has been copied.

To determine the staring location of the window in memory, we need to calculate the byte displacement from the start of screen memory, 131072. When a window is defined, it is given an origin, (XO,YO), which is the displacement from the pixel origin, the upper left hand corner. From this we can calculate the position of the first byte of the window. The byte displacement in the pixels per word, and then multiplied by 2, to get the number of bytes, INT(XO/8)*2. Similarly, the byte displacement in the y-direction would be YO*128 (128=512/8*2, total number of pixels per row divided by the number of pixels per word, times 2 to adjust for the number of bytes). Therefore, the starting position is 131072 + INT(XO/8)*2 + YO*128. Lines 14 to 19 in listing A performs the necessary calculations except XO is entered as the number of words and needs only to be converted to the number of bytes. The conversion of XO into the number of words is done in SuperBASIC before the routine is executed.

Lines 14 to 19 do not look exactly like the calculation above because, some techniques have been used to increase the speed of the calculation. When converting XO into bytes, instead of directly multiplying by 2, a logical shift is done. Whenever a number needs to be multiplied or divided by a power of 2 it can always be done in assembly language by shifting left or right, respectively, by the power. This shifting is also done when YO needs to be multiplied by 128. Instead it is shifted to the left by 7.

Steps 2) and 3) of the algorithm can easily be accomplished by two nested FOR loops or their equivalent in assembly language. The loops would cover the

complete dimensions of the window, (XX, YY). In listing A, lines 22 to 27 set up the initial values for the two loops labeled REPY and REPX (REPY2 & REPX2 for saving the screen. The discussion will only cover the window recalling portion because, saving and recalling are identical except for the direction of storage. See listing A.) One is subtracted from both XX and YY, the looping variables which are initially the dimensions of the window, because of the looping technique used.

The assembly language instruction DBxx is used for the main loop. It is a typical decrement and branch instruction with a slight twist. First, it keeps decrementing its control register until it equals negative one, which means one less than the actual number of times through the loop needs to be stored in the control register, and then control drops through to the next instruction. The twist to this instruction is the 'xx' part. It allows the testing of certain conditions and if met, passing execution to the next instruction regardless of the count. In our routine, we do not need this ability hence, we have it check the carry flag which is always reset after a MOVE instruction which is directly preceding the DBCS instruction. The next couple of instructions convert XX and YY into the number of displacement bytes of the appropriate directions. Finally, the value of the last byte position is calculated to determine when the entire window has been copied.

The main part of the routine is in lines 34 to 45. First, the counter for REPX is determined, which is half of XX since words will be copied instead of bytes, and stored. Next the loop is entered and a row of pixels is recalled word by word, lines 37 to 39. Indirect addressing with post increment is used to accomplish this task. The post increment will automatically position the pointer, SPOS, to the next word position in that row. The saved word is copied into a temporary storage register, line 37, and then copied to the screen, line 38. The counter is decrement, checked and branching taken appropriately, line 39.

Once out of the loop, the memory position of the first byte on the next row is calculated, lines 41 to 42. This is accomplished by subtracting XX from the current screen position, POS, which puts the pointer back on the first word of that row. Next, one hundred twenty-six is added to position the pointer on the first word of the next row. This value is compared to the actual last byte value of the window in line 44 and if greater than, then the window has been completely copied and the routine terminates, line 45. The only thing left to discuss about the assembly language part of the routine is lines 29 and 30. These two lines branch to the proper portion of the routine depending on the value in register D1, called TEMP in the listing. On D1=0, the window is saved and, on D1=1, the window is recalled.

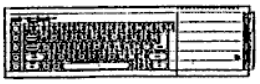
To use the swapping routine, listing A must be typed in and assembled or listing B must be typed in and ran. Listing A is an assembly language version of the routine. Listing B is a SuperBASIC program which will produce the same machine language file as listing A would after being assembled. The machine language file which must be loaded into memory with the following SuperBASIC commands, where XO,YO are the origin in pixels, XX,YY are the dimensions in pixels, XO_w,YO_w are the origin in words, and XX_w,YY_w are the dimensions in words of the window that will be swapped.

```
XO_w = INT(XO/8 - .5): IF XO_w < 0 THEN XO_w = 0
YO_w = YO - 1: IF YO_w < 0 THEN YO_w = 0
XX_w = INT(XX/8 +.5)+2: IF XO_w + XX_w >64 THEN XX_W=64-XO_w
YY_w = YO_w + 1: IF YO_w + YY_w >256 THEN YO_w = 256-YO_w
swap = RESPR(100 + YY_w*XX_w*2)
LBYTES {file name}, swap
```

The preliminary variable declarations usually only need to be computed once at the beginning of the program. The additions and subtractions of .5, 1 and 2 are needed to ensure the entire window will be saved if the window is not on the 8-pixel boundaries that the routine requires. The variables are in terms of the number of words because the routine needs the origin and position in that form.

Once loaded, to save the screen underneath the window execute the instruction: CALL swap,0,XO_w,YO_w,XX_w,YY_w. This executes the assembly language routine and puts 0, XO_w, YO_w, XX_w, YY_w into registers D1 through D5, respectively, which will cause the window area to be saved. Now, you can do whatever is needed in the window. When done, the window area can be restored with: CALL swap,1,XO_w,YO_w,XX_w,YY_w.

We now have a way to swap between windows that can be used in any program. Many copies of the routine can be used to have swapping among several windows. In the final window defining program, two windows will be swapped. One will have constantly changing origin and dimensions, which is why these values are left in the CALL instruction. The only limit on the number of windows being swapped is the amount of free memory.



In the next segment a useful alternative to the standard INPUT function of integers will be introduced. It will be written as a SuperBASIC function with plenty of explanations. Other functions will be given that will determine potentially needed information about a given window.

LISTING A.

```
 1: MEMORY    EQU     131072          Start of screen memory
 2: T128      EQU     7               Shift amount to multiply by 128
 3: T2        EQU     1               Shift amount to multiply by 2
 4: *
 5: POS       EQUR    A0              Current position reading
 6: SPOS      EQUR    A1              Current position saving to
 7: XO        EQUR    D2              Origin of window to save (1 to 64)
 8: YO        EQUR    D3                ------------------     (1 to 256)
 9: XX        EQUR    D4              Width of window to save
10: YY        EQUR    D5              Height of window to save
11: INC       EQUR    D6              Looping control register
12: TEMP      EQUR    D1              Byte being copied
13: *
14:           LEA.L   STORE,SPOS      Get start of memory to save to
15:           MOVE.L  #MEMORY,POS     Get start of screen memory
16:           LSL.L   #T2,XO          Times 2 to get the number of bytes
17:           ADDA.L  XO,POS          Add to start of screen memory
18:           LSL.L   #T128,YO        Times 128 to get number of bytes
19:           ADDA.L  YO,POS          Add to start of screen memory
20: *                                 POS now is at first byte of memory
21: *                                   to save.
22:           SUBQ.L  #1,YY           Setting up looping controls
23:           SUBQ.L  #1,XX                       :
24:           LSL.L   #T128,YY                    :
25:           LSL.L   #T2,XX                      :
26:           ADD.L   XX,YY                       :
27:           ADD.L   POS,YY          YY is pos. of last byte to save
28: *
29:          CMP.L    #0,TEMP        Test to see if saving or recalling
30:          BEQ      REPY2          Screen.(temp=0,save:temp=1,recall)
31: *
32: * Recall screen
33: *
34: REPY      MOVE.L  XX,INC          Number of bytes per line
35:           LSR.L   #T2,INC         Copying words so divide by 2
36: REPX
37:           MOVE.W  (SPOS)+,TEMP    Get saved word
38:           MOVE.W  TEMP,(POS)+     Put saved word on the screen
39:           DBCS    INC,REPX        Decrement counter and loop
40: *                                   appropriately
41:           SUBA.L  XX,POS          Done with line - Get position for
42:           ADDA.L  #126,POS          beginning of next line
43:
44:           CMP.L   POS,YY          Check to see if done
45:           BGE     REPY            Branch if not done
46:
47:           BRA.S   ENDING          Done so goto end
48: *
49: * Save screen
50: *
51: REPY2     MOVE.L  XX,INC          Number of bytes per line
52:           LSR.L   #T2,INC         Copying words so divide by 2
53: REPX2
54:           MOVE.W  (POS)+,TEMP     Get screen word
55:           MOVE.W  TEMP,(SPOS)+    Save word in memory
56:           DBCS    INC,REPX2       Decrement counter and loop
57: *                                   appropriately
58:           SUBA.L  XX,POS          Done with line - Get position for
59:           ADDA.L  #126,POS          beginning of next line
60:
61:           CMP.L   POS,YY          Check to see if done
62:           BGE     REPY2           Branch if not done
63:
64: ENDING    MOVEQ   #0,D0           No errors
```

```
65:           RTS                         Return
66:
67: STORE     DS.W      1                 Start of screen save region
68:
69:           END
```

LISTING B

```
10 OPEN_NEW #5, {file name} :REMark Replace {file name} with your file name
20 FOR I=1 TO 100
25    READ byte%
30    PRINT #5,CHR$(byte%);
40 END FOR i
45 PRINT #5,\
50 CLOSE #5
1000 DATA 67,250,0,96,32,124,0,2,0,128,227
1010 DATA 138,209,194,239,139,209,195,83,133,83,132
1020 DATA 239,141,227,140,218,132,218,136,178,188,0
1030 DATA 0,0,0,103,0,0,30,44,4,226,142
1040 DATA 50,25,48,193,85,206,255,250,145,196,209
1050 DATA 252,0,0,0,126,186,136,108,0,255,232
1060 DATA 96,26,44,4,226,142,50,24,50,193,85
1070 DATA 206,255,250,145,196,209,252,0,0,0,126
1080 DATA 186,136,108,0,255,232,112,0,78,117,0
1090 DATA 0
```

Editor notes: David McCall begins the first of a series in Update for Advanced Programming the Sinclair QL. David's outstanding tutorial on Machine Code and Super Basic programming will continue with three more installments. Each installment will contain 'stand alone' programming that can be keyed in and used. The four ensembles of programming may be merged into one operating program. This four article series will be followed by tutorials covering other major subjects. The series will become a valuable and permanent reference for those who are seeking to become profficient in programming the Sinclair QL.

****

EMSoft

Software for the QL that really works

We market software for business, financial and technical applications to do real (tax deductible) work. Prices range from $9.95 for QLUTter, a resident boot utility to $2850.00 for a site license for CAM MASTER, a very friendly CAD program to design circular cams and automatically generate the CNC code to a vertical mill controller.

We create much of our own software but we market software by other programmers. Please call/write for details about our arrangements.

New software in our catalogue includes RECIPE, PC-FOUR by PSION and the latest versions of TRUST_FUND, MAILBAG and QLAND_LORD. And we still have DBTutor, a flash-card program to teach yourself anything you need to know, and DBProgs with 5 useful Archive utilities. We have The_Transfer_V3, for HP compatible laser printing of DTP files and a range of other useful, hard-working software for the QL.

We will have TAX-I-QL/89 when the time comes. If you haven't tried it, we have TAX-I-QL/88 for $10.00 (+$2.00 on microcartridge) thru October 1, 1989. Try it on your 1988 return. If you like it, return the media with your invoice for full credit toward TAX-I-QL/89.

Generally, prices (post paid) are under US$25.00, but like CAM MASTER, some are a bit more. Send for our catalogue which explains things in detail. There is also a $9.95 demo disk for CAM MASTER.

EMSoft, P.O. Box 8763, Boston MA 02114 (617) 889-0830

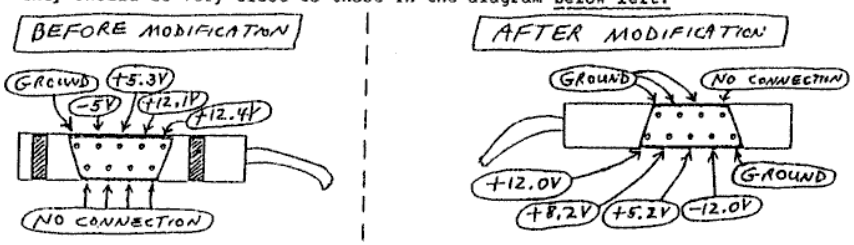
Wanted: Useful programs for the QL. Any topic that does real work; structural analysis, spreadsheet templates, stock market analysis, educational software. Royalties up to 85%. Contact EMSoft, Box 8763, Boston, MA 02114 or call (617) 889-0830.

HOW TO MODIFY A COLECO ADAM POWER SUPPLY FOR YOUR Q.L. By: Dan Elliott

Completion of this project requires the following:

An ACCURATE DVM, low temp. soldering iron, solder sucker, x-acto knife, 140 watt soldering gun, diagonal cutters, needle nose pliers, drill, a 7912 regulator chip, an assortment of resistors, a 9 pin male "D" connector, some 22 gauge insulated wire, and a little patience.

First plug in your ADAM power supply and test the output voltages. They should be very close to those in the diagram below left:



Next, unplug the power supply and remove the 4 screws that hold the heat sink to the chassis, and the 6 screws that hold the circuit board to the chassis. Now you can lift the circuit board with heat sink from the chassis to make the mods. Using the soldering gun, remove the red wires from the circuit board at E3 and E4. Remove the green wires from E8 and E7. Install the red wires in E8 and E7, and the green wires in E3 and E4. Next, with the soldering iron and solder sucker, unsolder U4 (79M05). Remove the screw holding it and remove the regulator. Install the 7912 regulator at U4, fasten and solder it. Remove R21 and replace it with a 12K resistor. Install a 2K shunt resistor across R8.

Cover the transformer with a piece of cardboard. Lay the circuit board on top of the cardboard and plug in the power supply. Measure the voltages at the end of the gray cable. If they do not agree with the voltages in the diagram above right, you will have to experiment with resistors at R21 for +8.2V, or R8 for +5.2V, or R32 for +12.0V.

Do not proceed with this modification until these voltages are correct.

Reassemble the power supply using the screws you removed earlier. With a Groung lug and a screw and nut, solder and fasten the blue, violet and bare wires to the chassis. With diagonal cutters and x-acto knife, remove the two rectangular alignment pins from the power output connector. Scrape off the letters "TOP" from the cable end, as this surface will become the bottom when you plug it into the Q.L. later on.

PART II - Q.L. MODIFICATION FOR ADAM POWER SUPPLY OPTION

Open up your Q.L. and lay the keyboard assembly aside. Use a 9 pin "D" connector as a template to mark the spot for mounting the power jack for the modified ADAM power supply. The ideal location is through the back of the Q.L. case just to the left of the mother board. Center the connector with narrow side facing up. Mark the back of the case with felt tip pen. Use these marks as a guide for drilling out the mounting holes. Use the X-acto knife to cut the hole in the shape of your "D" connector. When the hole is the correct shape, mark and drill the two holes for fastening the connector.
Next, strip and tin 5 pieces of 22 gauge insulated wire, each about 5 inches long. Solder them to pins 1 through 5 of the "D" connector.
Solder wire from pin #1 to left leg of C54. (+12V)
Solder wire from pin #2 to upper leg of C56. (+8V)
Solder wire from pin #3 to BUS to the right of C56. (+5V) - LOWER LEG
Solder wire from pin #4 to left leg of C58. (-12V)
Solder wire from pin #5 to right leg of C58. (GROUND)
Now mount the "D" connector using 4-40 or smaller screws and nuts. route the wires under the "D" connector and out of the way of the Q.L. expansion connector. Remove +5V regulator with heat sink from Q.L. Double check all of the wiring, hook up the modified power supply and fire it up! If you did it right, your Q.L. should initialize normally.

This modification will allow your Q.L. to run much cooler thus eliminating the crash problems associated with the inferior Q.L. power supply. If you still experience intermittent crash problems after doing this modification, then your mother board is probably defective.

The Unmodified ADAM power supplies are available from AMERICAN DESIGN COMPONENTS, and a few other surplus electronics dealers. If you have any trouble locating the parts or if you have other problems, write or call me and I will try to help you out.

DAN ELLIOTT Phone: (314) 739-1712 evenings, Sunday through Thursday.
RT 1, BOX 117
CABOOL, MO. 65689

"dprtr.B6"
A Universal Code Data File Printer.
For All TS-2068 Configurations
Donated to Public Domain, by Update Magazine.

```
5226 CLEAR : RANDOMIZE USR VAL "100": LOAD "ccle
ar.C1"CODE : RANDOMIZE USR VAL "100": LOAD "ca.C
1"SCREEN$ : PAUSE 0: LET z=CODE INKEY$-VAL "48":
: GO TO (z>VAL "2")*VAL "5226"+(z=VAL "2")*VAL "
5228"+(z=VAL "1")*VAL "5250"
5228 LET x=VAL "0": LET p=VAL "1": LET l=VAL "1"
: LET a=VAL "0": LET pl=VAL "56": LET c=VAL "64"
: RANDOMIZE USR VAL "100": LOAD "clnfo.C1"SCREEN
$ : PAUSE VAL "0"
5230 RANDOMIZE USR VAL "100": LOAD "cclr.C1"CODE
 : CLS : INPUT "Insert Data Disk, then ENTER DD#
";dd: RANDOMIZE USR VAL "100": GO TO dd: RANDOMI
ZE USR VAL "100": CAT ".C©",: INPUT "Key in Code
 File + EXT";w$: RANDOMIZE USR VAL "100": LOAD w
$CODE VAL "50000"
5232 INPUT "Install Program Disk, then       ENTE
R Disk #";dd: RANDOMIZE USR VAL "100": GO TO dd:
 INPUT "Input Line length";c: RANDOMIZE USR VAL
"100": POKE VAL "16090",(c-VAL "1"): CLS : INPUT
 "Print at TAB?";tab: RANDOMIZE USR VAL "100": P
OKE VAL "16094",tab: LPRINT : INPUT "Start print
 at Line#?";x: LET l=x: FOR g=VAL "1" TO x: LPRI
NT : NEXT g: INPUT "Lines per page?";pl: INPUT "
Page #?";p
5234 LET b=VAL "0": FOR n=50000 TO 65535: LET a=
PEEK n: IF a=0 THEN  GO TO 5242
5236 IF a=13 THEN  LPRINT '
5238 IF a>31 THEN  LPRINT CHR$ a;: LET b=b+1: IF
 b>63 THEN  LET b=0: LET L=L+1: IF L>c THEN  LPR
INT ''TAB 32;p: LET L=0: LET p=p+1: LPRINT : OUT
 127,12: LPRINT : BEEP 1,20: PAUSE 0
5240 NEXT n
5242 FOR n=L TO c: LPRINT : NEXT n: LPRINT TAB c
/VAL "2";p: OUT VAL "127",VAL "12": GO TO VAL "5
290"
5250 STOP
5290 CLS : PRINT AT VAL "10",VAL "0";"<1> Print
 another File"''"<2> QUIT": PAUSE VAL "0": LET z=
CODE INKEY$-VAL "48": GO TO (z>VAL "1")*VAL "530
0"+(z<VAL "2")*VAL "5230"
```

Fig. 1
ca.C1 SCREEN$

```
      FILE PRINTER Info

This program prints out DATA
Files Directly from Disk.

Either Sequential Files, or
CODE FILES as saved by Mscript
or TASWI may be elected.

_____________________________

SELECT Your Choice:

[1] Print Sequential Files

[2] Print Code File
```

Fig. 2
clnfo.C1 Screen$

```
      Code Printer Info

1. This program prints out CODE
   Files (Mscript- TASWI, etc)

2. You must Guess first for the
   Line Length. (Try 64)

3. Watch the print out.

4. IF NOT a formatted Line, then
   BREAK. Count the characters
   to find the Line length.

5. Insert a new PAGE, then OFF
   and ON the printer, LPRINT.

6. Then Type <GO TO 5230> ENTER.

Please make a note of (5230)
ENTER to Start.
```

The Universal Code Data Printer:

This neat program will work with any TS-2068 configuration, including Cassette, or Disk Systems. (with modifications). The program is designed to be a MERGE IN program for the new DAISY.B6 software, but it operates independantly, with RUN or GO TO 5226. When MERGED to any Basic program, it provides the capability to PRINT-OUT data files previously SAVED as ASCII Files in Code. Some examples of such files are "Mscript Files, Taswi Files, and files saved via Telecom Modem". For the Daisy.B6 software this "dprtr.B6" program is MERGED via a menu elective to "steal and print" data files created with other software. (Sneaky, eh?) Well, what the heck! Most TS-2068 users have one of the small word processors, either Mscript or Tasword, and many swap data files back and forth on either cassette or disk. Now Smart Text users can print out these files at will. And the Daisy.B6 program does it via menu elective.

Line #5226 starts right off with some intangables to be explained. 1. The program ASSUMES that a LKDOS INITIALIZATION has taken place, the type of interface poked to adr 16096, and line feed need poked to adr 16092. For other DOS systems, change the programming to initialize the DOS with printing parameters. For some this will involve adding a line ahead of $5226 to load the prcode and initialize it.

2. Still at line 5226, A "BLANK" Code program "cclear.C1" is loaded to CLEAR the area in memory from adr 50000 to 65367. To get this code program on disk or cassette, Turn ON the Computer and type < SAVE "cclear" CODE 50000,15367 >. This saves 15367 "zeroes", not worth too much for any purpose but to clear out that range of addresses when needed. We will use this space in memory later for loading all sorts of Code Data files to be printed, one file after another until the computer room overflows with paper.

3. Still at line 5226, we load in a Information Screen with <LOAD "ca.C1"SCREEN$. Refer to fig 1 for a print out of the screen. You can omit this in the program and just go on to the next line. By the way, the selection <1> is not included here. That "Print Sequential Files" routine will be added in the October issue of Update (!)! I can remember to do it!.

4. Another info screen is booted in with line #5228. Figure 2 shows this "clnfo.C1"SCREEN$ file. 5. At line #5230 we LOAD the cclear.C1 file of zeroes to clear out any code that may be hiding in the needed area 50000-65367. Then the disk directory is gotten and a prompt placed at bottom to input the code file name. Incidentally the "©" is my printer's character for the UP ARROW on the H Key. The Up Arrow with LKDOS is a wild card that gets all ".C" files for the directory. Continuing- The file name input is LOADED with a starting place of adr 50000. This limits the file length to 15367 total characters, enough for most code files. But if you want to allow for larger files, you can change the load-in address to a lower address in memory.

5. Line #5232 gets back the Program Disk and begins prompts for print-out. Since we DONT KNOW what line length the creator of the code file used, we have to guess. A smart guess would be "64". A few seconds later we will find out

whether the guess hit the mark. If the lines print with wrap-around words, we BREAK after a few lines have printed , and examine the printed lines. Count the characters in the lines and you can find the formatted line length. Then TYPE GO TO 5230. This time you can enter the proper line length and the print-out will be true to its prepared format.

6. Lines 5234 through 5340 uses the N counter to peek out the code characters and print them. Taswi files have padded in spaces (code 32) to format lines. Since this program is in BASIC, a line of spaces causes a short pause in the first couple of lines. After that the printer's character buffer begins to store characters and work ahead of the printer, thus the time for subsequent line spacing is absorbed. On the other hand, Mscript line spacing is gotten by the insertion of ENTER (chr Code 13) at the end of each line. Line #5236 takes care of Mr. Mscript. Other Ascii Files may use a combination of these techniques. Text files that contain PRINTER CODED Command Characters may need to be EDITED if the character codes are printable ones, as "/", "°", "*", etc. You can EDIT LINE 5238 and add within the "IF THEN" statement "IF a>13 OR a <123 THEN"-----. This will exclude any Un-Printable character codes from being processed.

7. Line #5238 contains programming to affect a Page END, providing a page number and a FORM FEED. You may remove the "PAUSE 0" if using a Dot Matrix printer. The pause is to allow the insertion of a cut sheet of paper. 8. Line 5242 is the clean-up programming after all code has been printed. The N counter is used again to space down to the page end and print the last page number. 9. Line #5250 is where we will start the next programming to print out Sequential Files (in the next issue of Update). 10. Line # 5290 allows you to go back and print another code file, or QUIT. The QUIT line (5300) is hiding in the DAISY.B6 program. Change the GO TO to whatever line number you need for Quitting. The following lines are used by Daisy.B6 to first MERGE this "dprtr program", then "DELETE it when QUIT is elected":

```
5220 CLEAR: RANDOMIZE USR VAL "100": MERGE "dprtr.B6"

5300 DELETE VAL "5226",VAL "5290": RANDOMIZE USR VAL "100": POKE VAL "16090", VAL "136": RANDOMIZE USR VAL "100": POKE VAL "16094", VAL "0": CLEAR: RANDOMIZE USR VAL "100": MERGE "varsv.B6": DIM K$(VAL "8"): DIM v$(VAL "1866"): GO TO VAL "9990"
```

The above is given for an insight into the Daisy.B6 manipulations of the MERGE function. First we DELETE the "dprtr.B6" program, THEN RESET the LKDOS EPROM. Then we MERGE in the VARIABLE SAVE LOAD program. Then we set up the K$ and V$ parameters to accommodate a VARIABLE FILE LOAD of the initialization vars file "01876.C2". GO TO 9990 affects the vars file load to re-initialize the Daisy.B6 software. (The Variable File SAVE LOAD program was given in a previous issue of Update). See you next issue with the SEQ FILE Printer. -BJ_

** ANNCUNCEMENTS **

HOST for 1990 Sinclair Conference??

Update received a letter from SMUG, 5052 N. 91st St., Milwaukee, WI 53225 (Bill Heberlin). Bill says that the SMUG group is interested in hosting a Sinclair Computer Fest for 1990. SMUG asks that all Dealers, or other persons, who are interested in participating in a 1990 Computer Fest to contact them. (To SMUG: Send your news releases to Update and we will help publicise.)

Public Domain Libraries Available

1. Cassette or Disk for TS-2068: The Ontario User Group: George Chambers, 14 Richome Court, Scarborough, Ontario M1K 1Y1, Canada. Put a 30 cent stamp on your letter from U.S. and inclose a buck for return mailing. (no U.S. Stamps or coin). Inquire for price of library.

2. CATS Library: TS-2068 Public Domain library on cassette, Oliger-LKDOS-FD-68 Disk. CATS, Box 457, Fairfax Station, VA 22039. Send $2.00 for information package.

3. SINCUS Library: TS-2068 Public Domain library on cassette, Oliger Disk, LKDOS Disk. Send $2.00 for information package to: SINCUS, John Colona, 28 Guilfoyle Ave, Binghampton, NY 13903.

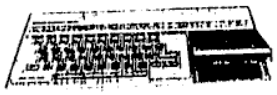
4. VISTA Library: TS-1000 and TS-2068 Public Domain Library on either Cassette or LKDOS Disk (LKDOS for TS-1000 also). Send $2.00 for information package to: Vista, Tony Willing, Box 199, Vashon, WA 98070.

5. ISTUG Library: TS-1000 and TS-2068 Public Domain Library: Cassette and Oliger Disk. Send $2.00 for info to: ISTUG Frank Davis, 513 East Main, Peru, IN 46970. Or call 317-473-8031

Subscriber Ads (For SALE)

Large list of good used TS-1000 and QL software and hardware. SASE or Compuserve E-MAIL 747,3432. Letters to Carlos Lytle, 1064 Worthington Woods Blv., Worthington, OH 43085. Or call 614-846-3543.

I have an extensive hoard of TS-2068 and TS-1000 software and books to sell. Send SASE for list. Marinus Heuseveldt, 3955 Hallmark Cir., Lantana, FL 33462

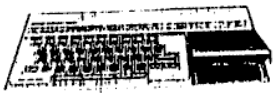


The FD-66 Update

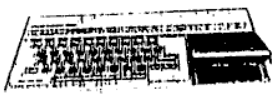
By: Syd Wyncoop

Last issue I left you with a promise of some routines to convert between Sinclair/Timex mode and CP/M mode on the FD-68. As the routines are quite long and I write my code to be self-documenting, I will give the routines first. Here is the CP/M to Sinclair routine:

```
;CP/M to Sinclair File Converter
;===============================
;
;This works by bank switching.  The file to be converted will be
;placed in the TPA and then transferred to chunks 6 & 7, in the
;home bank.  This places a file size restriction of 16K on the
;transferred file however, this is large enough to accomodate
;most machine code programs.  The command to do this is:
;
;                    CPM2SIN filename.ext
;
;After the transfer, all chunks of the home bank are switched in
;and the Sinclair ROM editing loop is entered, via a Basic NEW
;command (the transfered file resides above RAMTOP).  The user
;can then save the file as bytes, with the following command:
;
;              SAVE "filename"CODE 49152,16384
;
;As written, the CP/M file will always be moved to 49152 (chunk
;6) and is limited to but, may be shorter than 16384.  You
;should use the true file size, if known, in order to save
;SAVEing time.
;
;It should be noted that 16K is the practical limit to the file
;size to transfer, as R/PM uses chunk 4 (8000h) for the 64
;column display routines and chunk 5 (AA00h) for stashing the
;CCP for warm-booting.
;
;Variables
;---------
;These equates are standard variables that are part of a file
;that I start all assembly source files with.  It saves a lot of
;typing.
;
CPM             EQU  0000h      ;CP/M Warmstart address
BDOS            EQU  0005h      ;entry point for CP/M functions
BELL            EQU    07h      ;ASCII bell (beep) character
PRTSTRING       EQU    09h      ;CP/M function
TAB             EQU    09h      ;ASCII tab character
LF              EQU    0Ah      ;ASCII line feed character
CR              EQU    0Dh      ;ASCII carriage return character
OPENFILE        EQU    0Fh      ;CP/M function
CLOSEFILE       EQU    10h      ;CP/M function
READSEQ         EQU    14h      ;CP/M function
SETDMA          EQU    1Ah      ;CP/M function
FCB             EQU  005Ch      ;CP/M file control block
FILESIZE        EQU  006Bh      ;CP/M file parameters
HORZSELECT      EQU   0F4h      ;Timex hardware port
TPA             EQU  0100h      ;CP/M program area start
DMABUFFER       EQU  0500h      ;start of our transfer file area
NEW             EQU  0D62h      ;Timex ROM routine
RASP            EQU  5C38h      ;Home Bank - Timex variable
TIMEXVARS       EQU  5C3Ah      ;Home Bank - Timex variable area
RAMTOP          EQU  5CB2h      ;Home Bank - Timex variable
PRAMTOP         EQU  5CB4h      ;Home Bank - Timex variable
CHUNK6          EQU 0C000h      ;start address for memory chunk 6
MOTOROFF        EQU 0EF06h      ;CP/M BDOS routine
;
          ORG  TPA
;
;We start here.  First, we need to check whether the filename of
;the file to move has been supplied on the command line.  If
;not, we exit with an error message which gives the proper
;syntax to use.  If a filename was supplied, we try to open the
;file.  If unable to open the file,.we assume it does not exist
;and we exit with an appropriate error message.  Otherwise, we
;assume all is well and proceed to transfer the specified file
;to the Home bank of memory.
;
START:    LD DE,FCB + 1         ;point to filename
          LD A,(DE)             ;get first letter of filename
          CP " "                ;spaces mean no name supplied
          JR Z,SYNTAX           ;give syntax error message
```



```
            DEC DE              ;point to FCB
            LD C,OPENFILE       ;get function number
            CALL BDOS           ;try to open file
            INC A               ;check for error code FFh
            JR Z,NOTFOUND       ;exit if file not found
            LD DE,DMABUFFER     ;else, preset DMA address

;
;A filename was given and we were able to open the file.  We now
;need to read the entire file into the TPA.
;
READFILE:   PUSH DE             ;save DMA pointer
            LD C,SETDMA         ;get function number
            CALL BDOS           ;go set DMA
            LD DE,FCB           ;point to filename
            LD C,READSEQ        ;get function number
            CALL BDOS           ;read the file
            POP DE              ;retrieve DMA pointer
            AND A               ;check for EOF
            JR NZ,READDONE      ;done if EOF found
            LD HL,0080h         ;offset to next DMA
            ADD HL,DE           ;calculate new DMA
            EX DE,HL            ;put DMA back into DE
            JR READFILE         ;loop to keep reading

;
;This is the error message for an incorrect start to program.
;
SYNTAX:     CALL PRTMSG         ;put msg address on stack
            DEFB BELL, CR, LF
            DEFM 'ERROR!  No filename supplied.'
            DEFB BELL, CR, LF, LF
            DEFM 'Usage:  CPM2SIN filename.ext'
            DEFB CR, LF, LF, '$'

;
;This is the error message for a filename which was supplied but
;cannot be opened or cannot be found.
;
NOTFOUND:   CALL PRTMSG         ;put msg address on stack
            DEFB BELL, CR, LF
            DEFM 'ERROR!  File not found.'
            DEFB BELL, CR, LF, LF, '$'

;
;This is a subroutine to call BDOS to print any ASCII string
;which ends with a $.  It assumes the start address of the
;string to be printed has been put on the stack.
;
PRTMSG:     POP DE              ;retrieve message address
            LD C,PRTSTRING      ;get function number
            CALL BDOS           ;print error message
            JP CPM              ;return to CP/M

;
;This routine finishes the read operation by closing the file,
;turning off the disk motor and determining the file size to
;move to chunks 6 & 7 in the Home bank.
;
READDONE:   LD DE,FCB           ;point to filename
            LD C,CLOSEFILE      ;get function number
            CALL BDOS           ;close the file
            CALL MOTOROFF       ;turn off disk drive motor
            LD A,(FILESIZE)     ;retrieve number of records
            LD B,A              ;and put it in counter
            LD DE,0080h         ;set record size
            LD HL,0000h         ;clear accumulator
SIZELOOP:   ADD HL,DE           ;multiply FILESIZE * 128
            DJNZ SIZELOOP       ;until multiplication is done
            PUSH HL             ;place the result
            POP BC              ;into BC for block move

;
;We now have the specified file in the TPA and can proceed to
;transfer it to chunks 6 & 7, in the home bank of memory
;
TRANSFER:   DI                  ;first, no interrupts
            LD A,3Fh            ;reset bits 6 & 7 for home bank
            OUT (HORZSELECT),A  ;enable chunks 6 & 7
            LD HL,DMABUFFER     ;start of block to move
            LD DE,CHUNK6        ;start of destination block
            LDIR                ;do move

;
;If all went well, the file has been transferred and we are back
;in CP/M.  We need a routine to reset the Timex registers and
;jump back into the home ROM.  First, we move our routine to the
;Home bank, jump to it and after some minor housekeeping, we use
;the Home Rom's NEW routine to clear all memory other than
;chunks 6 & 7.
;
RESTORETIMEX:   LD A,0DFh           ;select chunk 5
                OUT (HORZSELECT),A  ;enable home bank chunk 5
```



```
                LD HL,JUMPTIMEX      ;set-up for block move
                LD DE,0A000h
                LD BC,DONE - JUMPTIMEX
                LDIR                 ;move it home bank
                JP 0A000h            ;go to home bank
;
;The following code is moved to the Home Bank, Chunk 5, and is
;run from there.
;
JUMPTIMEX:      LD A,3Fh             ;Timex initial values
                LD I,A
                IM 1
                LD IY,TIMEXVARS
                XOR A                ;select all chunks for home
                OUT (HORZSELECT),A   ;bank
                LD HL,0FFFFh
                LD (PRAMTOP),HL      ;set PRAMTOP
                LD H,0BFh            ;set RAMTOP value to protect
                LD (RAMTOP),HL       ;moved file at C000h
                LD BC,40h            ;length of raspberry warning
                LD (RASP),BC
                JP NEW               ;go to Timex ROM
;
DONE:           NOP                  ;place marker for assembler
;
                END
```

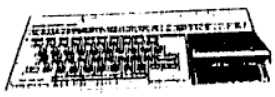
The above program does not re-enable the Eprom code for a clean and elegant re-entry to Sinclair mode with the FD-68 disk code enabled but, it works adequately enough for ocassional use. Another solution would be to redefine one of the logical drives from MD3 format to FD-68 format and use CP/M's BDOS to write the file directly. However, that is more than I needed for this demonstration.

A practical use for these programs? I have not found a text editor that works properly in CP/M mode on the FD-68 therefore, I continue to use MScript to write my assembly language files and I port the files to CP/M for assembly and debugging. I then port the finished object code back to the Timex mode.

The reason for doing this is that CP/M has much better tools for assembly language programming. I always use the Z80ASM assembler by SLR Systems, Butler, Pa, (telephone 412-282-0864) and the Z8E debugger, by Rick Surwilo. Z8E is public domain and supplied with the FD-68 or can be found in the SIG/M library.

The program for transfering files from Sinclair to CP/M is:

```
;Sinclair to CP/M File Converter
;===============================
;
;This works by bank switching.  The file to be converted must
;first be stored in the Home Bank, Chucks 6 & 7, while in Timex
;mode.  CP/M is then warm-booted by placing your CP/M boot disk
;into disk drive A and issuing the command:
;
;                'CAT "X.BUT",'
;
;This program is the companion file that switches in the Home
;Bank chunks from CP/M and moves the file to the TPA (check the
;January 1989 issue for definitions).  The Dook Bank chucks are
;then re-enabled, in order to re-establish CP/M and this routine
;is exited.
;
;The user can then issue a CP/M 'SAVE xx Filename.Ext' command
;to place the file on disk in CP/M format.  This could also be
;done by calling the BDOS routines to write the file to disk
;prior to exiting the program but, was not added here in the
;name of simplicity.  Review the READFILE routine in the CPM2SIN
;program which is described above to see the technique.
;
;Variables
;---------
CPM             EQU  0000h
BDOS            EQU  0005h
TPA             EQU  0100h
BANK6           EQU 0C000h
VIDMODE         EQU   0FFh
HORZSELECT      EQU   0F4h
HORZSTASH       EQU  6100h          ;CP/M bank chuck storage
;
```



```
            ORG  TPA
;
START:      DI                          ;for safety
            PUSH AF                     ;and save register & flags
            LD HL,START2                ;start of this program
            LD DE,6000h                 ;new location for program
            LD BC,FINISH - START2       ;length of program
            LDIR                        ;relocate this program
            JP 6000h                    ;goto relocated program

;
START2:     IN A,(HORZSELECT)           ;read horz select register
            LD (HORZSTASH),A            ;and save register contents
            LD A,3Fh                    ;select 6 & 7 for home bank
            OUT (HORZSELECT),A          ;enable chunks 6 & 7
            LD HL,BANK6                 ;start of block to move
            LD DE,TPA                   ;start of destination block
            LD BC,4000h                 ;length of chunks 6 & 7
            LDIR                        ;do move
            LD A,(HORZSTASH)            ;retrieve register contents
            OUT (HORZSELECT),A          ;restore CP/M chunks
            POP AF                      ;restore CP/M flags
            EI                          ;ok to interrupt
            JP CPM                      ;return to CP/M
FINISH:     NOP                         ;place marker for assembler
;
            END
```

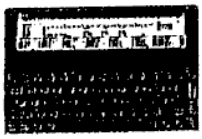
In case it is not clear, both of these programs run from CP/M and have the obvious names of CPM2SIN.COM and SIN2CPM.COM, respectively. The bulk of the demonstrations are given in the first program, CPM2SIN.

I'm not sure at this point what we will discuss next time. Send me some ideas or you may be sorry!

Editor's comments: Syd, this is to you, but also to other FD-68 experts. O.K., you ask for it! Here are some comments to help you plan for future columns. The Aerco FD-68 users have suffered from a lack of information support. You are the singular exception. The FD-68 is a powerful DOS but lacking in Extensions for common usage. For one, the .CHR extension is sorely missed for every day computing. Also, the inability to change the Disk selection pointer locks us into our existing disk configuration unless the EPROM is sent for re-programming (and dont hold your breath while waiting). Update believes that help is needed in the form of RAM based utilities to open up the 'missing extensions' to broaden the FD-68 DOS capabilities. We also need a quick source for EPROM re-programming, and a way to change our drive configurations and use BASIC to access the new config.

Update has broken ground to point out a serious danger of operating the FD-68 powered from the TS-2068. Others knew about this danger but kept quiet about it. Aerco once recommended a method of using the disk drive power supply to power the FD-68, so they recognized the problem early on. Their solution was not a solution because of the noise of disk switching power supplies. We suspect that many FD-68 users have left us because of TS-6068 destruction caused by the overloaded power supply when the FD-68 was in use. Dan Elliott has an inexpensive fix for this but he says that only a few have sent their FD-68's for the external power mod.

For some un-explained reason the Aerco FD-68 users have not benefited from an on-going publication of common use utilities as have other DOS system users. LKDOS users and Oliger DOS users have a plethora of disk utilities to load into RAM to do many tasks. These are in public domain, given by authors apart from the DOS manufacturer. Most of the FD-68 utilities that are available are proprietary and those of us who have them and could provide expansions are reluctant to do so because of infringement. The Aerco Printer Code has undergone several improvements which are not widely known about. The use of FD-68 extra memory is a dark and mysterous subject. Many FD-68 users have from 64K to 256K of extra RAM but few have ever used the extra RAM because of lack of programming information. So there is plenty to write about if one assumes (correctly) that most FD-68 users need useful utilities and help in the form of programming tutorials that are at the beginner and intermediate levels.
Update has had a continuing appeal for articles on the FD-68. Syd Wyncoop is the only FD-68 writer to come forth. We appreciate! _BJ-



A QL AUTO-LOADER

Bob Hartung, 2416 N. Co. Line Rd., Huntertown, IN 46748

By using the storage device for conversion of directory information to data which can be loaded back into array ct$(), this adaptation of the CAT_PRINT routine provides a menu-loader similar to the one for the TS2068 SAFE DOS user. It is written for MDV1 as the primary source drive but may be adapted for use on MDV2 and probably a DOS as well, although I do not have one to test that out.

On initialization, the routine stops with a WHEN ERRor prompt to create catalog-data file cat_dat. Once this file exists, when booted up the routine will run through a menu of all files in cat_dat by keying any character except (upper case) N. If N is keyed from the menu, cat_dat will be updated with the current directory on the drive selected.

MENULOAD

```
10 PAPER 0: INK 7: CLS: PAPER #0,7: INK #0,0: CLS #0: dr
ive$='mdv1_'
20 WHEN ERRor : PRINT #0;'RUN 240 to save DIR, then RUN'
: STOP: END WHEN
30 DIM ct$(100,12)
40 OPEN_IN #7,mdv1_cat_dat
50 INPUT #7,line$: PRINT 'CATALOG-';line$
60 FOR x = 0 TO 500
70    INPUT #7,ct$(x)
80    IF EOF(#7): LET F1=x: GO TO 100: END IF
90  END FOR x
100 CLOSE #7
110 PRINT #0\\'Key any CHR to scroll ENTER to load'\'N f
or NEW CATALOG'
120 A$='              '
130 LET F=1: x=7: Y=2: C=INT(F1/18): DIF=INT ((F1/18-C)*
18+.4): row=1: lop=17
140 LET Q$='p': IT=1: IF lop>=F1 THEN lop=F1: GO TO 160:
 END IF
150 FOR I=1 TO C: FOR M=0 TO lop: AT row+M,7: PRINT A$:
AT row+M,8: PRINT ct$(IT): LET IT=IT+1: NEXT M: GO SUB 1
90: NEXT I: FOR I=1 TO 19: AT I,7: PRINT A$: NEXT I: IF
NOT DIF THEN GO TO 140
160 IF F THEN FOR M=0 TO DIF-1: AT row+M,7: PRINT A$: AT
 row+M,8: PRINT ct$(IT): LET IT=IT+1: NEXT M: IF lop>=F1
 THEN F=0: END IF : END IF
170 IF NOT F THEN IT=F1+1: END IF
180 GO SUB 190: GO TO 140
190 FOR L=0 TO M: AT row+L,7: PRINT ')': FOR K=1 TO 5: N
EXT K: IF Q$()'' THEN FOR K=1 TO 100: NEXT K: LET Q$='':
 END IF
200 LET A=CODE(INKEY$): IF A=0 THEN LET Q$='p': GO TO 20
0: END IF
210 IF A=78 THEN GO SUB 240: GO TO 10
220 IF A=10 THEN GO TO 230: ELSE AT row+L,7: PRINT ' ':
NEXT L: RETurn : END IF
230 LET POS=IT+L-M-1: drive$=drive$ & ct$(POS): LRUN dri
ve$
240 CLS #0: INPUT #0; 'CATALOG drive no. ';drive
250 drive$='mdv' & drive & '_'
260 DELETE drive$ & 'cat_dat': OPEN_NEW #7,drive$ & 'cat
_dat'
270 DIR #7,drive$: CLOSE #7
280 RETurn
290 DEFine PROCedure sv: SAVE mdv1_BOOT: END DEFine sv
```

Z88 BATTERY REPLACEMENT

A few days ago I decided to do some work on a little Z88 program that I started over a month ago. I had used "QL Link" to store the un-finished Z88 program in a QL Disk. I opened the filing cabinet drawer and removed the Z88 (It was filed under Z.). I punched the Shift Keys to bring the Z88 on line, but alas, the Z88 was as dead as a duck. I thought, "No wonder. Its been a month since I've used it". Rummaging around, I found a 6 pack of fresh AA Cell Batteries. I never can come out even on pre-packaged things, can you? I flipped the dead Z88 over to its un-dignified position and removed the dead batts.

About that time the tea kettle whistled, so I made a cupa instant coffee and returned to install the new batteries. With my stubby fingers thats kinda like giving an emema to a mouse. A bit of fumbling got them in there.

Now the Z88 was back to its happy position, and I touch the two shift keys. Would you believe that up popped to screen the last data file that I had been working on a month ago? Well, it did!

I said out loud, "Well hello Z88, I thought you were dead. You and ole Rip Van Wrinkle must be related". Maybe I imagined it, but I'll swear that I heard a chirp in reply. And, the Wife said, "Talking to yourself again, eh"?

A file cabinet makes a nice place to stash the Z88 in a letter size folder, better than a drawer where one may stuff something on top of it.

I loaded the Z88 program to the QL memory from disk and then to the Z88 with QL Link. I still haven't finished the program, but I'll tell you what it is. Its the "Dbase-1" program given in this issue. But there is a problem. I dont know how to install and call the Machine Language utilities yet with the Z88. Can anyone help?

Actually that will just be an exercise in programming, as the Z88 "Diary" does almost everything that Dbase-1 does, and more. --BJ__

S.L.S. Again (Sudden Lock-up Syndrome)

Since getting the two Update QLs equipped with the external power supply by Dan Elliott about 4 months ago S.L.S. hadn't grabbed me - until yesterday. I was typing away on a Quill text and my right 4th pinkie split a hit of the K and L keys. Apparantly the QL (or Quill) intrepeted this as Omega and then "Fritz", and both being diehard Republicans, Fritz was turned to "Pffut, we aint workin fer no Democrat", and both quit on the spot. No coaxing or apology would un-freeze the screen. This leads me to believe that either the QL and its keyboard read, or the QUILL character read, one or the other, has a bug in thar that pops up and sez, "I warn't programmed to read split key hits and whut you done is greek to me". But one lock up in 4 months isn't enough to fuss over. RESET works.

The CABLE COLUMN

WOOD AND WIND COMPUTING

The Cable C Issue Disk

Bill Cable : RR 2 Box 92 : Cornish, NH 03745 : 603 675 2218

In the last Cable Column (April 89) I discussed how to program in ARCHIVE including a guide to using the built-in ARCHIVE editor. Please keep that article handy when entering the ARCHIVE programs described in this column. I have heard from several UPDATE readers who are indeed trying ARCHIVE for the first time. If anyone gets stumped feel free to send me a SASE with a brief description of your problem and I will try to help you. I will also try to cover problems or requests of general interest in this column.

ARCHIVE features

ARCHIVE has a language and like SUPERBASIC, you either have to write your own ARCHIVE programs or use programs others have written to benefit from its features. Why use ARCHIVE instead of SUPERBASIC? The ARCHIVE language is designed to handle information in both string and numeric form. It has powerful built in commands that are difficult to duplicate with SUPERBASIC. If you have applications that involve information handling such as address lists, financial records, index files of any sort you may find ARCHIVE to be the best language available on the QL for your application.

Besides its excellent language and built in editor it has an interesting feature that makes it stand apart from most databases. It stores its records exactly as they are instead of making them all the same length. This makes it very efficient in storing data. Most databases require you to specify the length of each field that makes up a record. This means you have to allow for the longest expected value. If you allow 20 characters for the city field in an address database then each record has 20 characters set aside for city name. If the average city length is 10 then half the space is wasted and if a city comes up that is 22 characters long it has to be truncated. ARCHIVE on the other hand, uses only the amount of space each city takes up. This expains why you can get 600 addresses with a complex 25 field address database on one cartridge. If all those fields had to have space alloted to them even if they were not used it would take 3 cartridges to hold them. This makes it possible to design your databases with enough fields to cover all situations needed.

The down side of using flexible length records is that you don't know where your records are unless you have a pointer list to tell you where each one is. That is why ARCHIVE must have its opened files closed properly. If that is not done the pointer list may be incomplete and you will get the message 'wrong file type' when you try to access it. Its a matter of opinion whether having the flexible length records is worth the vulnerability. I would like ARCHIVE even if it didn't have this unique feature. Careful backing up takes care of most of the vulnerability. People who get complacent about their software/hardware and don't make backups are taking great risks no matter what machine they have and what software they use.

FILER - A progam for file handling in ARCHIVE

FILER is the name of a group of ARCHIVE procedures listed below which should help you with your ARCHIVE file tasks. It allows for easy copying, deleting, formatting from within ARCHIVE. Always be careful when formatting because anything already on what you format will be lost. Use FILER to back up your databases by making copies.

There is a lot of detail packed into the code that may not be clear at first. Notice how the 'error' statement is used to check for possible errors and protect the procedures from being halted by errors. I make use of the ARCHIVE screen reversing character,chr(26). Whenever you print chr(26) the screen will reverse ink,paper until you print chr(26) again. The only global variable is ans$ which I use to pass values back to calling procedures. The getkey() function allows for single key

input much of the time. A lot of screen positioning is used to give the appearance of windows. The procedures box and get_device may be useful in your own programs. Box creates a cleared area on the screen in normal or reverse color. Get_device will prompt somewhere on the screen for a device and return its name in ans$. Look up unfamiliar commands and functions in your QL manual.

Type in FILER by starting ARCHIVE and entering the editor. After typing it in save it. You can run it by typing : **run '()filer'<ENTER>** where ()=device or typing **filer<ENTER>** if it is already loaded. You can jump to FILER from a run statement in your own program and then jump back to your program from FILER. You can merge it with your own program. It should make backing up from within ARCHIVE easy for you. You must close your database before copying it. If you want to copy a file using the same name you can just press enter when prompted for the target name. In the listing below the bold is for emphasis.

```
proc box;line1,col1,line2,col2,reverse
  local j
  if reverse: print chr(26): endif : let j=line1
  while j<line2+1: print at j,col1; tab col2;: let j=j+1: endwhile
  if reverse: print chr(26): endif
  endproc
proc do_copy;i$,j$
  print at 20,52;"(doing copy now)";: backup i$ as j$
  endproc
proc do_delete;i$
  print at 23,52;"(deleteing "+i$;: kill i$
  endproc
proc do_format;i$
  print at 22,52;"(formatting "+i$+")";: format i$
  endproc
proc do_run;i$,j$
  print at 22,52;"(pr"+j$+" run "+i$+")";
  if j$="g": run i$: else : run object i$: endif
  endproc
proc err_check;l,c,m$
  if errnum()
    print at l,c;"Err = ";errnum(); at l+1,c;m$;" failed. Press <ENTER>";
    input m$;: endif
  endproc



proc filer
  local sd$,td$,td$,tf$,c$,i$,j$
  mode 0: while 1
    cls : print at 0,29;"F I L E R";:p_26:get_device;2,20,"Which Device?"
    if aus$="":p_26: cls : return : endif : let sd$=aus$: let c$="?"
    print at 12,22;"(directory of "+sd$+")";
    while c$<>"e":p_26: dir sd$:p_26
      box;13,50,24,80,0: print at 13,51;"FILER MENU FOR ";upper(sd$);
      print at 14,53;"c   copy a file"
      print at 15,53;"d   delete a file"
      print at 16,53;"g   run a prg program "
      print at 17,53;"o   run a pro program"
      print at 18,53;"f   format medium in ";sd$;
      print at 19,53;"e   exit FILER MENU"
      print at 20,51;"Press key : ";:p_26: print " ";
      let c$=lower(getkey()): print c$;" ";:p_26
      if c$="c":get_device;13,50,"Copy from "+sd$+" to where?": let td$=ans$
        let sf$="?": while sf$<>"" and td$<>""
          if sd$=td$:p_26: dir sd$:p_26: endif :box;13,50,23,80,0
          print at 13,58;"FILE COPY";
          print at 14,56;sd$;" to ";td$; at 16,52;"copy ";sd$;
          input sf$: if sf$<>"": print at 18,52;"To   ";td$;: input tf$;
            if sd$<>td$ and tf$="": let tf$=sf$: print tf$: endif
            if tf$<>"": error do_copy;sd$+sf$,td$+tf$:err_check;22,51,"Copy"
              endif : endif : endwhile : endif
      if c$="d"
```

```
        print at 22,52;"Delete : ";sd$;: input sf$
        if sf$<>"": error do_delete;sd$+sf$: endif : endif
      if c$="g" or c$="o"
        print at 21,52;"Run : ";sd$;: input sf$: if sf$<>""
          error do_run;sd$+sf$,c$:err_check;23,52,"Run": endif : endif
      if c$="f"
        print at 20,51;"Format ";sd$;" - y/n : ";: input i$;: if lower(i$)="y"
          input at 21,52;"Give a name : ";i$;
          if i$<>"": error do_format;sd$+i$:err_check;23,51,"Format": endif
          endif : endif : endwhile :p_26: endwhile
  endproc
proc get_device;l,c,m$
  local i$,j$
  box;l,c,l+11,c+30,0: print at l,c+8;"DEVICE LIST";
  print at l+2,c+3;"1. mdv1_      5. ram1_"; at l+3,c+3;"2. mdv2_      6. ram2_";
  print at l+4,c+3;"3. flp1_      7. Other"; at l+5,c+3;"4. flp2_      8. Done";
  print at l+7,c+1;m$;: let ans$=""
  while ans$=""
    print at l+8,c;rept(" ",30);
    print at l+8,c+3;"Press key [1 to 8] : ";:p_26: print " ";
    let i$=getkey(): print i$;" ";:p_26
    if i$="1": let ans$="mdv1_": endif
    if i$="2": let ans$="mdv2_": endif : if i$="3": let ans$="flp1_": endif
    if i$="4": let ans$="flp2_": endif : if i$="5": let ans$="ram1_": endif
    if i$="6": let ans$="ram2_": endif : if i$="8": let ans$="": return : endif
    if i$="7": input at l+9,c+2;"Device name : ";j$;
      if len(j$)=5: if j$(5)="_": let ans$=j$: endif : endif
      if ans$=""
        input at l+10,c+2;"Not a device [<ENTER>] : ";j$;
        box;l+9,c,l+10,c+30,0: endif : endif
  endwhile
  endproc
proc p_26
  print chr(26);
  endproc
proc start
  filer
  endproc
```


WOOD AND WIND COMPUTING

Bill Cable : RR 2 Box 92 : Cornish, NH 03745 : 603 675 2218

Business and Educational Software for the QL

DBTutor : An ARCHIVE based learning shell. This program allows users to set up databases to learn any subjects they wish. It works well with any topic that can be typed in. Many interesting drills can be done to help learn the material including : flashcard, type in answer, reverse question and answer, score keeping, print missed questions, selected by subject, keys, or word occurence. Parents can set up drills for their children. Students can use it to learn languages or any school subject. QL users can use it to help learn computer related topics. The material can also be printed out in test or complete form. Long after the material has been entered it can be quickly reviewed. DBTutor uses the power of ARCHIVE without requiring the user to know anything about ARCHIVE. It is all menu driven. Comes with a detailed 25 page manual and many sample databases. I have a version that works very well on unexpanded QLs also. Order directy or send SASE for a list of other software available. (5 1/4 DSDD disk $24.95 or 2 MDV $28.95)

PLOTTING the GRAPH of an EQUATION

The T/S 2068 is an ideal computer for plotting graphs of algebraic or trigonometric equations, for display purposes. Although its number of pixels available for high-resolution art work is a little less than the number in much more expensive computers, the Timex greatly offsets this minor deficiency for several important reasons. One is because the T/S plots in the first quadrant, which means that the origin is in the lower, left-hand corner making X measurements positive to the right and Y measurements positive up. This agrees with well-established, mathematical practice. The Apple IIe, a widely distributed computer in our schools, draws in an awkward fourth quadrant and measures Y values positive in the down direction. Another reason is the capability of the Timex to be able to mix high-resolution graphics with text in any arrangement on the screen and in any order within a program. The Apple, on the other hand, provides only a four-line window at the bottom of the screen for text. This hinders the need to place text in or around the drawing itself.

At this point it might be appropriate to admit to readers that some knowledge of geometry would be helpful in digesting this material, but it is not absolutely necessary. Most of this discussion will only involve simple arithmetic and measurement.

Perhaps the best way to explain how to plot a graph on the T/S computer is to go thru the usual steps one encounters using a sample equation. For this purpose consider the equation..... Y=1/X.

Starting with an eye inspection, one can readily see that Y is negative for negative values of X, and that Y is positive for all positive values of X. Also note that Y becomes infinite when X equals zero. Thus the graph of this equation will lie in the first and third quadrants and its presentation will be enhanced by having its principal X- and Y-axes centered in the working area of the screen. So the first step is to relocate them from their left-hand and bottom positions.

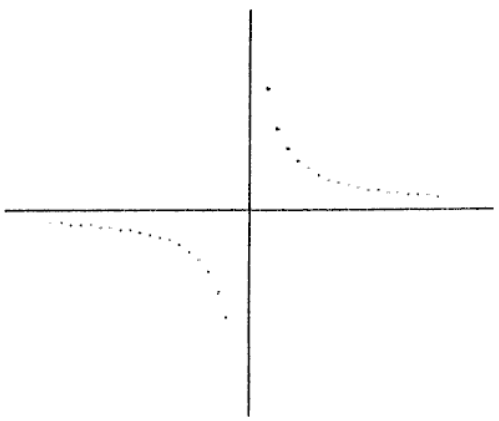
For reasons which will soon become apparent, the distance of translation for these axes will be expressed in units of rows and/or columns, converted to the corresponding number of pixels, there being 8 pixels in the width of one row or one column. Thus the shift to the right for the Y-axis will be called X0 and will have a value in this example of 16 columns times 8 pixels per column. The upward shift of the X-axis will be called Y0 and have a value of 11 rows times 8 pixels per row.

Our next concern is scale. We have to enlarge the ordinates (Y values) and the abscissa (X values) of the curve to see it when plotted. Otherwise the curve would appear as nothing more than a few smudges around the intersection of the relocated axes. This part takes some practice. Errors in judgement, however, are easy to correct using the procedure recommended herein. In this example we will arbitrarily let 50 pixels represent 1 unit of X, the horizontal scale, and 10 pixels to represent 1 unit of Y, the vertical scale. It works out best to use multiples of ten, i.e., 10, 20, 30, etc. We will refer to these scale factors as SX and SY in the computer program coming up.

The program starts off with the initial lines devoted to putting a frame around the working area of the screen, plotting the relocated axes, and defining the arbitrary values previously discussed. Then a single FOR-NEXT loop is used to draw the graph itself. You may save yourself time by omitting the REM statements in your own keyboard entry. See nx Pg

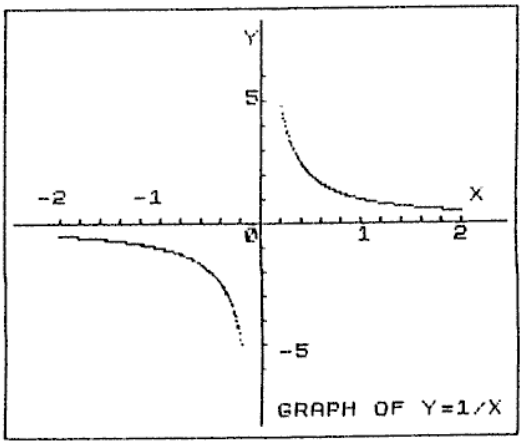
Profile of a Subscriber: Can you top this? Fredrich Hill, Woburn, Mass. is a TS-2068 user and subscriber. Fred was born in 1904, before more than 5% of the homes in America had Electricity, before the Model-T Ford, before Radio in 99% of homes, before refrigerators, before vacuum cleaners, before bikinis, before 'almost anything' that we take for granted today. Only a few principle streets in Boston were paved in 1904, those with bricks. Every city had livery stables and the down-town streets had center divides for parking horse and buggys. The principle job of city sanitation was the never ending accumulation of horse manure. Rains sometime exacerbated the problem. Two Wright brothers were preparing a contraption made of bicycle parts to attempt to fly, which most thought to be riciculuous. Quite a few States had not yet become States. Now Fredrich is a TS-2068 Computer Enthusiast. Salute!

```
  4 LET X0=8*16: LET Y0=8*11
  5 BORDER 6: REM ** DEFINES
                    WORKING AREA
 15 PLOT 5,Y0: DRAW 245,0
 20 PLOT X0,5: DRAW 0,165
 30 LET XS=50: LET YS=10

 40 REM ** PLOTTING THE GRAPH
100 FOR X=-2 TO +2 STEP 0.1:
REM ** ADJUST STEP LATER IF
      NECESSARY
110 LET Y=1/X: REM ** ORIGINAL
                  EQUATION
120 LET XX=XS*X+X0
130 LET YY=YS*Y+Y0
140 IF X>-.2 AND X<+.2 THEN NEX
T X: REM ** AVOID DIVISION BY 0
150 PLOT XX,YY: REM ** PLOT
                  ADJUSTED VALUES
160 NEXT X
```



After running this program, you can see that we assumed reasonably good values for scale, but that the curve needs more plotted points for it to stand out better. Do this now by changing the STEP value in line 100 to a smaller one. The only other work required to complete the picture is to add the scale values to the axes so that one can read corresponding X and Y values off the graph... and a title. All of that is done by the addition of lines 170 to 420. Again, you can skip the REM statements in your copy to lessen the work.

```
  2 REM ** GRAPH OF Y=1/X
  3 REM ** "C-59A", 9-22-87

  4 LET X0=8*16: LET Y0=8*11
  5 BORDER 6: REM ** DEFINES
                    WORKING AREA
 15 PLOT 5,Y0: DRAW 245,0
 20 PLOT X0,5: DRAW 0,165
 30 LET XS=50: LET YS=10

 40 REM ** PLOTTING THE GRAPH
100 FOR X=-2 TO +2 STEP 0.01:
REM ** ADJUST STEP LATER IF
      NECESSARY
110 LET Y=1/X: REM ** ORIGINAL
                  EQUATION
120 LET XX=XS*X+X0
130 LET YY=YS*Y+Y0
140 IF X>-.2 AND X<+.2 THEN NEX
T X: REM ** AVOID DIVISION BY 0
150 PLOT XX,YY: REM ** PLOT
                  ADJUSTED VALUES
160 NEXT X

170 REM ** FRAME THE DRAWING
180 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175

190 REM ** HORIZ. HASH MARKS
200 FOR N=X0-100 TO X0+100 STEP
10
210 PLOT N,Y0+1: PLOT N,Y0+2
220 NEXT N

290 REM ** VERT. HASH MARKS
300 FOR N=Y0-60 TO Y0+60 STEP 1
0
310 PLOT X0+1,N: PLOT X0+2,N
320 NEXT N

390 REM ** SCALE VALUES & TEXT
400 PRINT AT 9,2;"-2";TAB 8;"-1
";AT 9,29;"X";AT 11,15;"0";AT 11
,22;"1";TAB 28;"2"
410 PRINT AT 1,15;"Y";AT 4,15;"
5";AT 17,17;"-5"
420 PRINT AT 20,17;"GRAPH OF Y=
1/X"
```



Note that the graph is self-contained and that it has all of the information needed to stand alone.

In some future article we will explore how this concept of graph drawing can be applied to other mathematical curves besides the rectangular hyperbola (Y=1/X) which was used here.

Warren Fricke

A Fun Contest

Kilroy is the legendary sneaky Pete of World War II. "Kilroy was here" could be seen everywhere in little signs all over the USA, Canada, Europe and Asia. It is said that when Gen. McArthur stepped foot on Japan for the first time he saw the note "Kilroy was here". No one ever saw Kilroy in the present tense. He was always there "before". Hitler thought that Kilroy was a master Spy and had a general order out for his capture. Talk about computer viruses! Kilroy weaseled his way into the prges of Update. He was there and gone as usual. Find where he was just for the fun of it. The first letter received that tells" where Kilroy was" will receive honorable mention in the next issue.

>>>>>>>>>> QL PERKS & QUIRKS <<<<<<<<<<

QL Configurations ? ? ? ?

This blurb is intended to provide provocation to those who can answer some questions and enlighten us on the various Tool Kits for the Sinclar QL.

References are continually made to this tool kit or that ROM or that VERSION of the Sinclair QL. Most are in the dark as to what each tool kit provides. Now I know that this is asking a lot, but maybe someone will do it. Will someone please do an article for the Update readers that lists the various configurations of the QL and its various Tool Kits, with a brief on what the tool kits do? To start, I will duplicate the initialization screens of Update's two QL computers (below). This should put some to scratching their heads as both computers have had different "Hospital operations" by the leading QL Docktor in North America (Tom Bent). When looking at the initialization screens remember that there are no plug-in cartridges installed. These are the bare computer's report as the F1 screen comes on.

QL System #1:

Creative Code Works — Speed Screen Chip
Serial 39306 e 1989 — Simon N. Goodwin

Miracle Systems — V1.20 e 1984
Trump Card — 896K RAM
Care/Jump — Toolkit V2.11 e 1985

Now the next "pair" of screens are from Update's other QL. This spare QL has a slide switch installed on the rear deck for selection of tool kits in ROM. The screens will be given for each position of the slide switch.

QL Computer #2, slide switch Left Screen:

Miracle Systems Tool Kit V2.12 e 1985

Miracle Systems V1.20 e 1984
Trump Card 896K RAM

Care/QJUMP Tool Kit V2.11 e 1985

QL Computer #2, slide switch Right Screen:

Miracle Systems V1.20 e 1984

Trump Card 896K RAM

Care/QJUMP Toolkit V2.11 e 1985

The idea of having two QL Computers with such variety of tool kits was to provide for "Networking" of two QL computers so that all Tool Kit capabilities would be available to both computers as they are used in a "two computer network". (The EPROM for ROM replacement does not provide sufficient space to include Speed Screen and all Tools in one set.).

>> Some Quirks are not Quirks, but Perks <<

As above, there are many outstanding capabilities of the QL that are gnobilis only when adorned with literary recogniglon, upon which time previous obscurity of comment turns to profound epistiles for posterity. Since there are only some 2000 or fewer QL users in North America, and our experts number about the same as the population of extrovert aardvarks, we must politely plead for all bashful but knowledgable QL users to come out of the closet and share their knowledge. The rewards will be their elevation to guru status and profound appreciation. Some worthy subjects for enlightenment are:

NETWORKING: The most widely acclaimed capability of the QL (in early publicity) is its ability to network up to 64 computers hooked together. Most users are not interested in such a gang of QL's fighting over a printer or a disk drive, but the details of how to hook up "just two", and how to go about doing intelligent coordination, would be very interesting. If you've done it tell us how we can do it.

TRANSFERRING DATA AND PROGRAMS BETWEEN COMPUTERS:

We need to know how to affect a direct link up between two computers, and the computer commands to use to transfer. And, how about someone telling us how to get a data file and a program listing from our TS-2068 to our QL? Some have solved that problem by installing two phone lines and using modems on both computers, but thats about like moving the house to stretch a clothes line.

CAPTURING Files from DISK to Disassemble:

How does one List an EXEC file to paper to disassemble and examine?

How does one List Non-PSION Files, disk to paper?

All of the above are questions from subscribers. Others are asking for an article about String Concatenations and the use of Character Arrays, both being somewhat different than other computers.

>>> Current Inputs <<<

John Guergens gives us some more Quirks and Tips as well. Methinks that John has become the continuator of cognizables which are consumedly rarified but occasionally have stupifying terminations; himself being a cognoscente rather than a neophyte. John is a steady contributor and a QL consummate. we are grateful. Questions beget answers which procreate more questions, adinfinitum. So the title "QL QUIRKS" is changed to "QL PERKS and QUIRKS", more appropriate and less negative dont you think?

April 28, 1989

John E. Juergens
18 Bryce Canyon Way
Pacifica, CA 94044

Bill Jones
c/o UPDATE Magazine
1317 Stratford Avenue
Panama City, FL 32404

Dear Bill,

Your April issue arrived yesterday. As April proceeded past the 15th, began to think you forgot about out-putting UPDATE for this quarter.

MAiled APR 7 BJ

Re: Your "OPINION", p. 11: It's been so long since I have experienced a QLWP/QUILL SAVE to the default data device at about 2-1/2 to 3 pages of memorable prose - not since Curry Computer convinced me early on that a 512K expansion board was the solution to that and many other problems - that my recollection is a bit hazy. However, I am certain that our version of QLWP/QUILL did not SAVE without notice.

Pony EXPRESS BACK To USing HORSES?

I suspect that you, like myself on more than one occasion, had your head down pounding away at the keyboard and didn't notice the warning posted on the screen. My further recollection is that unless you immediately pressed the "secret" key mentioned in the warning any other, such as simply continuing to type away, would activate the SAVE to the default data device under the file_name def_tmp; in your case, MDV 2.

Long gALlop FLA TO CA

The situation seems to be that there are few choices at that point. There is no further RAM available to store additional text so one either manually does a SAVE, Zap's what's been done and continues on a clean page, so to speak, but postpones the bigger, inevitable problem of how to print out the sucker in one acceptable whole. Or let QLWP/QUILL handle the problem for you in a much better fashion than I have ever been able to come up with.

Personally, I think the QLWP/QUILL solution is pretty elegant: As you continue to add text, it automatically updates the document to def_tmp on the data default device. This allows either a print out of the whole document, automatically extracting text from def_tmp when needed, or an F3, SAVE under the file_name and device of your choice. The F3, SAVE, again automatically, incorporates what was in def_tmp and then deletes def_tmp from the default device cartridge/disk. A subsequent LOAD of the F3, SAVEd file, again automatically, creates a def_tmp file on the default device and SAVEs thereto the portion which won't fit into QLWP/QUILL allowing you to continue to add-to and finally PRINT the whole shebang without worry about tying pages together.

If one had only the standard 128K of RAM and an empty cartridge in the data default device (to catch the def_tmps) one effectively should have, at 5 sectors per page and 200 sectors per cartridge, about 40 typed pages available if the F3, SAVE is done to an empty cartridge in the non-default drive. Using a DSQD disk drive (LRUN CONFIG bas to change System, Data and Help default devices) should make 280+ pages available.

The disk scenario is probably sufficient for a novelette or doctoral dissertation (at least for the appendicies) and, unlike what happened to someone I heard about who was doing his dissertation on a hard disk and was about finished when the inevitable crash occurred - with 'nothing retrievable by the bye - the QL user has his/her's SAVEd to a reasonably secure medium.

about The Z88

Ever since the Z88 was introduced I have read each advertisement and comment with great curiosity as to how it works and what it will do better than its predecessors. Your exposition, p. 6, et seq, told me more about the Z88 than all of the others combined. So...some questions which I hope will generate responses of general interest.

P. 6, left, bottom col: "...on bottom screen, "End of Text.": If it didn't mean you were out of memory, what did it mean? You left me hanging. Me Too!

With the 128K RAMcard in place does it still give you an "End of Text" report at the point you were when it came on or does any RAMcard in place automatically expand the memory available for text typing? I guess the question is, put another way, is an installed RAMcard available only for file saving, a la a cartridge or disk or will one automatically make available more application RAM?

didn't go BACK To THAT Point.

If the file SAVEs are automatic upon switching to another application, what is the name of the file(s) SAVEd since you (apparently) could not assign one at the switching SAVE? Or, is one assigned at the time of entering the application and that becomes the filename under which the automatic SAVE occurs?

See Answers Appended

Were you actually able to do the column layout found in UPDATE from within PIPEDREAM and print it out as we see it?

Yep.

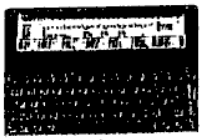
What are the "portable" uses for it that you have found or can envision?

MuchoGRande!

As the Z88's in/out is serial, how fast are the microdrive or disk up-and-downloads?

9600 Baud

<But Slower With ZN-DOS>



Other BBC. Basic programs + MC programs — as other computers

You mentioned that the Z88 "...provid(es) permanent SAVE of data or programs in EPROM." What "programs" are there to be SAVEd? Unless I misunderstand, aren't there really only 2 programs, PIPEDREADM and DIARY, (each of which may contain sufficient commands/functions to amount to 4 or so conventionally thought-of-as programs?) And, aren't they built-in? If so, what's to SAVE? Or, is one or the other, or both, capable of being programmed like QLDB/Archive?

How do you manage to get all of these not inexpensive purchases past Mrs. J?

Your comment, "Cambridge must have this Bank Switching scheme perfected for idiot operation.", prompts the following offering of a bit of wisdom attributed to David Robb and found in the latest issue of Astounding,Analog Science Fiction, Science Fact (for such a small mag it sure has a lot of names):

> "For every idiot-proof system devised, a new, improved idiot will arise to overcome it."

MR Robb, a Follower of Murphy's Law, No doubt!

For the QL Quirks Colum: Is our version of QLWP the only one which causes MDV 2 to grind on and on in an F3, SAVE, "overwrite?" (y)es situation? The first SAVE always takes a reasonable time. However, the subsequent ones go on interminably. My solution has been to SAVE to ramdisk and then F3, (O)ther, (F)iles, (B)ackup RAMn file_name to MDV2_file_name, ENTER, "overwrite?", (y)es which always results in a SAVE of reasonable time. There's an obvious alternate to this solution: Buy a disk drive!

TAPE Positioning?

Concerning our newly acquired disk drives: They arrived and worked well except for a quirk: FLP 2 drive will display a "not found" error report when loaded with a disk it did not format and exclusivply write to. A FLP2 disk may be read by FLP 1, be returned to FLP 2 and honored. One non-FLP 2 write to a FLP 2 disk and all kinship disowned - Talk about Copy Protection! FLP 1, on the other hand, accepts its own as well as one written to by FLP 2.

See Comments below.

When apprised of the situation the vendor could not assign a cause and suggested I return the drive and interface for checking-out. Good IDEA! TRY That First.

Surprizing, to me at least, was the discovery that fast as a floppy drive is, it takes longer to get a disk program up and running than mdvloading a cartridge to ram disk via QfLash on Eprom and executing from ram. Mdvloading provides a bonus of loading all of our printer_dats to RAMn allowing near-instaneous change-over to accomodate various printing devices.

? Un-believable!

Also, file copying, even with the Cumana wcopy (set of) command(s), is noticeably slower than the QfLash fbackup command to an MDV, if one is not constrained by the 100K/200 sector limit of a cartridge. However, beyond the two instances mentioned, it is disk ahead of MDV by a wide margin in terms of time consumed and sector availability.

?

While in SuperBasic the combination of the QfLash ramdisk and command set coupled with the Cumana wcopy and other commands uncovered by FLP_EXT is simply extraordinary for the occasionally necessary dog-work of moving disk files around, lining them up in a particular order, deleting and copying. Although my testing could not be considered exhaustive, I've found no conflicts to date.

Very truly yours,

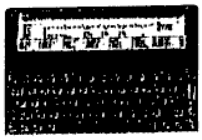
A few twits and quacks in reply

John Juergens comes in again with an intriguing letter having mixed questions and answers. John's letters remind me of one "Alexander P. Botts", a field representative of "Earthworm Tractor Co", and his letters to the home office. These letters ran for years in the (old) Saturday Evening Post. For those younger than these classics, the letters always described escapades where Alex's ineptitude caused horrendous results, such as a run-away goliath tractor in a cranberry bog where the competition's was mired down. The tractor mows down fences and barns and is chased all over the county until it runs out of fuel. Botts is fired because of the huge damage expense, always to be re-instated with salary increase when a multi million sale results from the un-intended demonstration of the superior power of the Earthworm Tractor. In this context I will answer some of the questions, defer others to the home office, and ask for those who have solutions to send them in.

1. The Z88's "End of Text" message was apparantly inserted into the machine to create conversation, with the premise that the more the Z88 is talked about the more there will be folks who will take a chance and buy one. The report does not stop the action, and if politely ignored one can continue operating but with the nagging worry that disaster may be enema (nt). So far I haven't realized what the report is trying to tell me. (Stupid answers always gets corrections).

2. The "Quill Auto Save" falls into the above category of stupid answers. You wouldn't believe how many wrote in and told me how good that feature is for those having only 128K systems. Me? I'm always swimming around in a bigger pool of memory that an IBM AT can handle and have never run across that Perk.

3. The Z88 AUTO SAVE: Boy, this is a nice feature! Say that you are working on a long term paper or a report that takes days to finish. Along comes a letter that requires an answer. You shift out of Pipe Dream, and the existing report that you are working on is automatically put into "suspended activities" with a title and time date. You then answer the letter and then pick up where you left



off with the long report. This takes a second or two for those who are proficient with the Z88, but a little longer for those of us who have the head bone crammed with TS-2068 and Sinclair QL stuff.

3. The Z88 Cartridge Banks: The Z88 has four banks (cartridge slots). Each bank can have 1 megabyte of memory. Cartridges are available now for 32K up to 512K of RAM capacity each. The 1 Mgbt cartridge is not yet available. The Z88 keeps the Data in RAM permanent even when the Z88 is OFF. Actually there is fourth Bank, "the Home Bank", which is the built-in memory of the Z88. The home bank is the smallest, having a total of 64K of combined ROM and RAM. The computer can operate in only one bank at a time, which is like saying that your car can only roll on one street at a time. But, one can easily turn from first street to 52nd Ave., and keep on trucking. In this case, the four banks act as "side streets" to store data and to operate programs. Shifting between banks of memory is via keyboard commands or by lines of programming in BBC Basic (or machine code programs). This is kinda like having three super fast internal Disk Drives to call upon to store data and programs. But the storage is in Chip Memory. Operating programs can have programming that switches to other banks of memory to retrieve or save data or, or to execute stored programs for their data product output to be used in the operating program. The two 'built in softwares', Pipedream and Diary, are fixed in ROM and always available for use. As these programs are operated the data that is created is protected. When one shifts OUT of the software to another Z88 function or program, the existing data is placed into "suspended activities" and held under a 'time date'. The suspended activities files remain protected until they are intentionally deleted, much in the same manner as Disk Files remain in a diskette until erased.

4. Getting past the Family Auditor (Wife): She has her on spending vices and we rationalize to each other about how we are always saving by our wise spending logic. I've already amortized the cost of my Z88 system by not drinking beer for five months. But once in awhile the plastic money tree sheds its fruits and causes a period of drought until the next ship arrives.

5. Z88 PRINTING: Yep. The review in the April issue was printed with the Z88 and the daisy wheel (centronics parallel) printer that I normally use with the QL. The connection was via the Z88 Parallel interface cable to the printer, exactly the same set-up as used with the QL. The Z88's printer driver program is in ROM, needing no extra software, and is set up via menu for different printer requirements. This is similiar to the 'configure' program for the QL's PSION software group.

6. Your disk drive problem indicates that the read write head is a bit out of alignment. The symptom is: It reads its own recorded programs but wont read programs recorded by another drive. Shop repair usually costs about $50 if you can find a capable shop. A new drive costs about $50 and you have temporary return rights. Trial and error tweaking sometimes works.

Cambridge Z88 NewsNotes™

INFORMATION CHANNELS: When UPDATE purchased the Z88 back in January 1989 three issues of "News Notes" were received. NewsNotes is a very informative two page User newsletter published by Cambridge North America. After not receiving follow on newsletters, a call was made to the telephone numbers given in the literature supplied with the Z88 computer. A recording was received "Sorry, but that telephone number has been DISCONNECTED". Other Z88 users may have reached a dead end with attempts to contact Cambridge North America. Follow on persistance resulted in finding the current telephone and postal addresses of Cambridge North America.

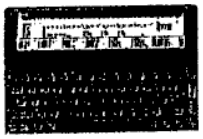
The Sales and Marketing office of Cambridge North America receives mail and telephone calls at: Cambridge North America, 750 West Algonquin, Arlington Heights, IL 60005. Telephones: 800 366 0088 and 312 593 0253. This is the appropriate office for calls and correspondance pertaining to Technical Information and publications. A helpful person there is Mr. Grant Kief. Mr. Kief promised to keep Update Magazine on their mailing list for all releases of User and Technical information about the Z88.

The Cambridge North America office for Warranty and repair is: Cambridge North America, 424 Cumberland Ave., Portland, ME 04104. Telephone number: 207 761 3700. This is also the Corporate Headquarters of Cambridge North America. This is the office to contact for Repair Service and for general business pertaining to Dealerships and Corporate business. Mr. Maurice Giguere is the "hands on" CEO.

The publication of Cambridge NewsNotes has been suspended during the period of office moves. Update Magazine hopes that this nice newsletter will be resumed. A suitable alternative would be the regular release of USER INFORMATION to Magazines that are willing to devote page space for USER SUPPORT. Such programming and technical information from the manufacturer is needed during the early life of a new computer while user support is developing. Once programming support materializes within the user group the magazines can fulfill the on-going needs for information support. If a sufficient flow of user information from Cambridge North America develops, Update Magazine will publish a regular column titled "Cambridge News Notes".

NEW Z88 Support: A recent development extends the versatility of the Z88. A Disk Drive system for the Z88 is now available from your dealer. This is a portable battery operated 3 1/2" disk system that matches the motif of the Z88 as a "carry around" computer system. Call your dealer.

The width of the Z88 market encompasses the whole computer industry. Traditional Sinclair users, such as the subscribers to Update, are keenly interested in the Z88 but represent only a significant fraction of the user group that is forming. This new computer also has high appeal to IBM users, Macintosh users, and to the general public for portability. It should be fun to pace this growing user group in the pages of Update Magazine and extend our horizions. -BJ_



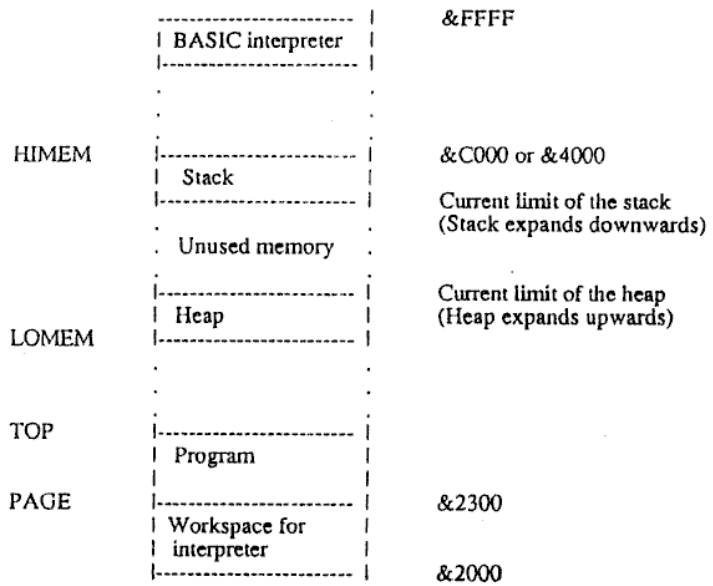
Organization of memory

The Z88 is based on a Z80 central processor running at 3.2876 MHz. The Z80 has a 16-bit address bus and can thus only directly access 64Kbytes of memory. Reconciling oneself to this total amount of memory would be very restrictive for a machine like the Z88 and in fact the Z88 can use up to 4Mbytes of memory. The decoding is handled by the 'BLINK' gate array, which sits between the CPU and memory. The CPU communicates via i/o ports, which address various BLINK registers.
The 4Mbyte physical address space is split up into 256 'banks', each of 16Kbytes. At any time, the Z80 can address 4 such banks. The 64K logical address space is notionally divided into four 16K 'segments', which are by convention numbered as follows:

Segment 0: Logical addresses &0000 - &3FFF
Segment 1: Logical addresses &4000 - &7FFF
Segment 2: Logical addresses &8000 - &BFFF
Segment 3: Logical addresses &C000 - &FFFF

Facilities are provided for a program to 'bind' each segment to any of the 256 banks. This means that a logical address falling within each segment will be translated into a physical address within the bank bound to that segment, the lower 14 bits of the logical address providing an offset within the bank.

BASIC's program/workspace is arranged in the following manner:

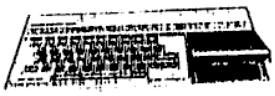


The Expanded and non-Expanded Machine

When a Z88 has 128K or more of RAM in slot 1 it becomes an expanded machine. The differences between the expanded and non-expanded are as follows:

Property	Expanded	Non-Expanded
Size of BASIC	40K	8K
Maximum map width	256 pixels	80 pixels
User characters	64	16 (but see below)
Value of EOF#-1	-1	0

Some users will want to use extra memory, for filing for example, without the burden of a 40K BASIC. If all your BASIC programs fit comfortably within the ordinary 8K it is unhelpful to have to carry the extra 32K around, thus using slot 2 or slot 3 to expand the memory does not expand the machine, but just increases memory size. (Note that using Slot 3 for RAM causes a very heavy power drain on the Z88, due to special hardware for programming EPROMs.) The unexpanded machine can use 64 user characters, but if an 80 pixel map is used the last 48 of these will be overwritten by map information when PipeDream is used. Reducing the map width to 64 pixels, or not using the map at all, allows for free use of all 64 user characters.



D..base-1

```
6800 DIM n$(5,31): LET n$(1)="Update Magazine": LET n$(2)="1317 Stratford Ave": LET n$(3)="Panam
a City, FL 32404": LET n$(4)="Bill Jones": LET n $(5)="Sincerely,"
6804 BORDER 0: PAPER 0: INK 7: CLS : PRINT AT 3, 6;"Dbase-1 Menu"'TAB 2;"<1> Create File"'TAB 2;
"<2> Add Records"'TAB 2;"<3> View or Print Recor ds"'TAB 2;"<4> Correct Records"'TAB 2;"<5> SAVE
 File"'TAB 2;"<6> LOAD File"'TAB 2;"<7> Sort an d Manage"'TAB 2;"<8> To Function Menu"'TAB 2;"<9
> Letterhead or Sign Off";#0;AT 0,0;"Touch Selec tion": PAUSE 0: LET z=CODE INKEY$-48: IF z>9 THE
N  GO TO 6800
6808 GO TO (Z<1 OR z>9)*6808+(Z=1)*6812+(z=2 OR z=4)*6824+(z=3)*7440+(Z=5)*7264+(Z=6)*7272+(Z=7)
*7020+(Z=8)*9999+(z=9)*7530
6812 CLS : PRINT AT 10,6; FLASH 1;"Confirm, STAR T NEW"; FLASH 0''TAB 6;"<y> Yes -or- <n> No": PA USE 0: LET z=CODE INKEY$-48: IF Z=2 THEN  GO TO
6804
6814 CLS : DIM o$(1,1): INPUT "Input Nbr of Reco rds";L1: INPUT "Nbr of Lines?";L2: INPUT "Chrs p er Line?";L3: IF L1*L2*L3+24*L2*L3+600> FREE  TH EN  BEEP .5,30: GO TO 6814
6818 DIM o$(L1,L2,L3): GO TO 6804
6824 CLS : INPUT "Start at Record #?";ms: INPUT "End Record $?";mx
6830 FOR o=ms TO mx
6834 IF z=4 THEN  GO TO 6850
6838 IF z=2 THEN  GO TO 6864
6840 NEXT o: IF o>mx THEN  GO TO 6804
6842 STOP
6850 PRINT AT 2,0;"Record # ";o': FOR n=1 TO L2: PRINT "Line ";n'o$(o,n)': NEXT n: PRINT ''"O.K. ? <1> Yes  <2> No  <3> Quit": PAUSE 0: LET q=COD E INKEY$-48: IF q=2 THEN  GO TO 6858
6854 IF q=3 THEN  GO TO 6804
6856 IF q=1 THEN  CLS : GO TO 6840
6858 INPUT "Enter Line Nbr to Correct";q1: INPUT  "Key In the new Line                ";y$: LET o$(o ,q1)=y$: CLS : GO TO 6852
6864 CLS : PRINT AT 2,0;"Record Nbr ";o'': FOR n =1 TO L2: PRINT AT 20,0;"Key In Line # ";n'"or z z to Quit";AT n*2+1,0;"": INPUT y$: IF y$="zz" T HEN  LET n=L2: LET o=mx: GO TO 6804
6868 LET o$(o,n)=y$
6872 PRINT "Line ";n'o$(o,n)'
6876 NEXT n
6880 GO TO 6840
7020 BORDER 0: INK 7: PAPER 0: CLS : PRINT AT 2, 0;"File Management Options"''"1> Remove Repeats" '"2> Remove Skips"'"3> Move Zip to line 4"''"4> Sort Data Base"'"5> Break Lg File to 4 Files
    and Save"'"6> Combine 4 Small Disk Files
  to Sort"''"7> Save a Sorted File"'"8> Load a l arge File"'"9> Clear Memory"''"0> Present Key Re cords                (Line 7 Key)"'"> View or Prin t Records"'"; > To Function Menu"; INPUT "Input C hoice";z$: LET z=CODE z$-48: IF z>11 THEN  GO TO  7020
7024 IF z>7 THEN  GO TO 7050
7028 ON ERR  GO TO 7032: IF LEN o$(1,1)>1 THEN ON ERR  GO TO 7036: STOP
7032 ON ERR  RESET : CLS : PRINT AT 10,0; FLASH 1;"No Data Present": BEEP 1,20: GO TO 7020
7036 ON ERR  RESET : GO SUB 7162
7050 GO TO (z=1)*7110+(z=2)*7132+(z=3)*7070+(z=4 )*7054+(z=5)*7240+(z=6)*7280+(z=7)*7264+(z=8)*72 72+(z=9)*7126+(z=10)*7440+(z=0)*7400+(z=11)*6804

7052 REM Sort menu
7054 CLS : PRINT AT 14,0;"There are ";L2;" lines  that can be  sorted,": INPUT "Input Line# to S ort";so: PRINT AT 18,0;"You may sort by FIRST GR OUP of  Line ";so;" or LAST GROUP:": INPUT "Inpu t <1> First or <2> Last";sol: IF sol<1 OR sol>2 OR so<1 OR so>L2 THEN  CLS : GO TO 7054
7058>LET b$="":GO TO (sol=1)*7364+(sol=2)*7294
7070 REM zip codes to line 4
7074 CLS : PRINT "Zip Codes to Line 4": FOR n=1 10 L5: IF o$(n,4)(1)=" " THEN  LET m$=o$(n,3)
```

**** The New "D.Base-1" Software for Sinclair Computers

An Issue Disk Program

"Dbase-1" is a new software for Sinclair Computers. The programming is done in TS-2068 simple Basic, which challenges programmers of the Cambridge Z88 to clone the programming in BBC Basic. Actually, Dbase-1 will be very easy to apply to both the QL and the Z88, requiring only rudimentary changes in syntax for counter operation, string slicing, and display control. Then of course the coding of the short Machine Code routines will be different for the QL and Z88. But why would emulation be desirable for these two advanced computers? I believe that an examination of the power and flexibility of Dbase-1 will be convincing enough.

Update Magazine again teams up with Don Lamen, Sincus. This issue Update presents the whole D.Base-1 program in BASIC. Then in the next issue, the sorting routines will be given in Machine Code by Don. By presenting the programming in this fashion, this issue becomes a detailed tutorial on Basic and the next issue will be a tutorial on Cloning Basic in Machine code. Actually the Basic program as given here works just fine for me. I'm kinda patient and the short wait for sorting doesn't upset me much and gives me time to THINK. This will be a lesson on "String and Character Array Management". We will slice, concatenate, truncate, swap array fields, use counters, nested counters, and tripple nested counters. Sinclair Computers have the ROM capability to do these things, so why not?

SINCLAIR COMPUTERS DO IT BETTER! The ONE MOST SUPERIOR ability of Sinclair Computers is their built in management of Character Arrays. Most other computers limit "String Content" to 255 characters. The TS-2068 allows over 4000 characters to be contained in a string, while the QL allows 32000 characters. The Z88 has similiar capability. Moreover, Sinclair Computers are rich in "string management" abilities. The Computer's ROM contains more management functions for character arrays than any other computer. To build a software that manages data "outside of the variable file" requires thousands of bytes of programming and error trapping. Dbase-1 takes advantage of the built in capabilities of Sinclair computers for string and character array management within the Variable File.

There exists many Data Base programs, each for special purpose. There are Mail Lists, Inventory Lists, Appointment calenders, Dairy herd milking records, Telephone lists, Daily Diary, etc. Some have print out capability and others do not. Some have Sorting and some do not. Why not "a Single Data Base Software" to use for all purposes? Dbase-1 is so very flexible that it can be used for any application that needs "multiple records". The user needs to learn only one software and then can shape the data base to fit any need. Fourteen SORTING Methods are provided, which is un-precedented. Several Print-Out methods are provided, including special print out for Labels, and Outlines. You can use Dbase-1 for virtually ANY type of listing purpose that you can conceive of.

DBASE-1 CONCEPTS: We make maximum use of the Computer's FREE MEMORY, and coordinate Disk Drive (Micodrive) to manage Data Base Files. The SORTING functions give the data base its flexibility for use for any purpose. The USER defines the Data Base Dimensions, thus allowing the RECORDS to be shaped for optimum use, by purpose. This allows the ONE software to create and manage files of many dimensions and applications. File dimensions (TS-2068) can vary from a set of 96 Mailing File records to 400 shorter inventory records, to fewer LONG records for other purposes. The QL's files in memory

```
7078 IF o$(n,4)(1)<>" " THEN  GO TO 7094
7082 IF m$(LEN m$)=" " THEN  LET m$=m$( TO LEN m
$-1): GO TO 7082
7086 FOR g=LEN m$ TO 1 STEP -1: IF m$(g)=" " THE
N  LET o$(n,4)=m$(g+1 TO ): LET o$(n,3)=o$(n,3)(
 TO g): GO TO 7094
7090 NEXT g
7094 IF n>=L5 THEN  GO TO 7020
7098 NEXT n
7102 GO TO 7020
7106 REM Elim repeats
7110 CLS : PRINT "Removing Repeats": FOR n=1 TO
L5: FOR y=L5 TO n+1 STEP -1: IF o$(y,1)=o$(n,1)
THEN  FOR g=1 TO L2: LET o$(y,g)="": NEXT g
7114 NEXT y
7118 NEXT n
7122 GO TO 7020
7126 DIM o$(1): DIM v$(1): DIM k$(1): DIM v$(1):
 GO TO 7020
7130 REM Eliminate Skips
7132 LET g=1: LET y=1: CLS : PRINT "Removing Ski
ps": FOR n=1 TO L1: LET x=0: IF o$(n,1)(1)=" " T
HEN  LET x=1: FOR g=2 TO L2: LET o$(n,g)="": NEX
T g: FOR y=n+1 TO L1: IF o$(y,1)(1)<>" " THEN  F
OR g=1 TO L2: LET o$(n,g)=o$(y,g): NEXT g: FOR g
=1 TO L2: LET o$(y,g)="": NEXT g: GO TO 7142
7138 IF x=1 THEN  NEXT y
7142 NEXT n
7146 GO TO 7020
7150 REM Save Zip Sort
7154 GO SUB 7256: INPUT "Zip File # ";z: LET z$=
"z"+STR$ z+".A$": RANDOMIZE USR 100: SAVE z$ DAT
A o$(): GO SUB 7232: GO TO 7020
7158 REM find array dimension #
7162 PRINT "FINDING DIMENSION OF O$"'': LET a=0:
 LET L4=0: LET L3=0: LET L2=0: LET L1=0: FOR n=1
 TO 400: ON ERR  GO TO 7170: IF LEN o$(1,1,1,n)>
1 AND LEN o$(2,2,2,n)>1 THEN  LET a=n
7166 NEXT n
7170 ON ERR  RESET : LET L4=n-1
7174 LET a=0: FOR n=1 TO 400: ON ERR  GO TO 7182
: IF LEN o$(1,1,n)>1 AND LEN o$(2,2,n)>1 THEN  L
ET a=n
7178 NEXT n
7182 ON ERR  RESET : LET L3=n-1:
7186 LET a=0: FOR n=1 TO 400: ON ERR  GO TO 7194
: IF LEN o$(1,n)>1 AND LEN o$(2,n)>1 THEN  LET a
=n
7190 NEXT n
7194 ON ERR  RESET : LET L2=n-1
7198 LET a=0: FOR n=1 TO 400: ON ERR  GO TO 7206
: IF LEN o$(n,1)>1 AND LEN o$(n,2)>1 THEN  LET a
=n
7202 NEXT n
7206 ON ERR  RESET : LET L1=n-1
7210 CLS : PRINT AT 10,0;"The Data Base Is O$(";
L1;": PRINT ("" AND L2<2)+"s"," AND L2>1);: IF L2>
1 THEN  PRINT L2;("" AND L3<2)+"," AND L3>1);:
IF L3>1 THEN  PRINT L3;
7214 PRINT ")"'': FOR n=1 TO L1: ON ERR  GO TO 7
218: IF o$(n,1)(1)<>" " THEN  LET xx=n
7216 NEXT n: ON ERR  GO TO 7218: STOP
7218 ON ERR  RESET : LET L5=xx
7220 PRINT "The last Filled Record Is: ";L5
7224 PRINT ''"Please Make Note and ENTER": PAUSE
 0: RETURN
7228 REM Disk Prompt
7232 CLS : INPUT "Re-Install Program DISK, and
  ENTER DRIVE # ";d: RANDOMIZE USR 100: GO TO d:
 RETURN
7236 REM Split/Save groups of 4
7240 CLS : INPUT "Are these Alpha or Zip Files?
  <1> Alpha - <2> Zip?";a: GO SUB 7256

7244 LET xx=1: FOR n=1 TO 96 STEP 24: DIM v$(24,
L2,L3): LET m$=("a"+STR$ s4+".A$" AND a<2)+("z"+
STR$ s4+".A$" AND a>1): FOR y=1 TO 24: FOR g=1 T
O L2: LET v$(y,g)=o$(xx,g): NEXT g: LET xx=xx+1:
 NEXT y: RANDOMIZE USR 100: SAVE m$ DATA v$(): L
ET s4=s4+1: NEXT n: GO SUB 7232: RANDOMIZE USR 1
00: CAT ".A$",: PRINT #0;AT 0,0;"Note the SAVED
files,"''"Enter": PAUSE 0: GO TO 7020
```

can be much larger, up to 2000 records of a mail list type of file (un-expanded QL), or 6000 or more smaller records.

MEMORY MANAGEMENT: The programming length of Dbase-1 is 12000 bytes, which leaves the TS-2068 having around 26000 bytes of FREE memory. We reserve a "buffer" in memory to perform disk file management, leaving 21000 bytes for data base generation. The QL and Z88 will have a much greater allocation for the data base. Now lets see how this 21000 bytes can be used "by menu elective". The data base is created by selection of "<1> Create" at the menu. A series of prompts comes on that allow you to "Shape the Data Base". The result of three inputs "Dimensions the character array "O$". Prompts ask for: "Number of Records", "Number of Lines", and "Line Length".

Suppose that you need a Mail File of four lines for the name and address, plus one line for the person's first name, plus two other lines for information about the person. That would be 7 lines for each RECORD. How long should each line be? Well, generally 31 characters is sufficient for the longest of address lines, so lets use 31. Now, how many records? Lets experiment and input 100. Then the three inputs would be: 100, 7, and 31. Line number 8614 would attempt to Dimension O$(100,7,31), for a file lenght of 100*7*31 bytes. But there is a trap there in line 8416 which catches inputs that are too large, thus reserving memory for the built in Buffer. The result would be the asking for new inputs. This is because 100*7*32 is too big.

Try 96 as the first input, then 7 and 31, and this set of inputs would be accepted. To continue with other examples: Suppose you want to create invoice records of 5 lines of 64 characters per line. Then 5*64=320 characters per record. You have 21000 available bytes, so you could create 21000/320=65 invoices. Suppose that you want to create an inventory file of records having 3 lines of 20 characters per line. Then each record would be only 60 characters. 21000/60=350 possible records. Yes, I know that you are thinking ahead and asking "What about the print out?". Several print-out formats are available at menu. Dbase-1, as it is given here, does not initialize a Printer Port Code. Add your printer port code load and initialization in a line number above 6800. The Dbase-1 issue disk will contain the printer initialization for the type of disk system that the user specifies.

THE PROGRAMMING: Line number 6800 initializes a small data base (N$) for your Letterhead and signature lines. Change the LETs in this line for your letterhead lines. The letterhead prints at center page when elected at menu. A menu option allows printing a single letterhead or multiple copies of Letterhead stationary. Also, menu options allow a "sign off" to be printed upon demand. The sign off is "Sincerely, your name". These print products were conceived for invoices. Then Line 6804 is the primary Menu, having 9 electives. Each of these electives are straight-forward. The data base is created from this menu. Line 6808 contains the branches (GO TO) for each of the menu electives. This is in boolean logic which compresses 10 "IF THEN GO TO" evaluations into one line and conserves program length.

Line # 6814 is important because it establishes three key variables which are used throughout the program. L1 is the number of records, L2 is the number of lines in each record, and L3 is the line length. During management of Disk Files, the "Find Dimension" routine at line 7162 operates after a file has been loaded from disk. The disk file is then in memory, and its dimensions are found and assigned to variables L1, L2, and L3. Also the "last filled record" of the file is assigned to variable L5. These four variables allow the management of three dimensioned character arrays of various dimensions. One warning becomes appropriate. The management system

```
7252 REM Disk prompt
7256 INPUT "Install DATA disk, then  Enter Disk
#";d: RANDOMIZE USR 100: GO TO d: RANDOMIZE USR
100: CAT ".A$",: RETURN
7260 REM Save File
7264 CLS : GO SUB 7256: INPUT "<1> Alpha Save- o
r <2> Zip Save ";az: INPUT "Enter File Nbr?";a:
LET m$=("A"+STR$ a AND az<2)+("Z"+STR$ a AND az>
1): LET m$=m$+".A$": RANDOMIZE USR 100: SAVE m$
DATA o$(): GO TO 7020
7268 REM Load Alpha File
7272 CLS : GO SUB 7256: INPUT "<1> Alpha File- <
2> Zip File";a: INPUT "Input File Nbr to Load ";
m$: LET m$=("A"+m$ AND a<2)+("Z"+m$ AND a>1): DI
M o$(1,1,1): DIM v$(1,1): LET m$=m$+".A$": RANDO
MIZE USR 100: LOAD m$ DATA o$(): GO SUB 7232: GO
 SUB 7162: GO TO 7020
7276 REM Merge 4 small files
7280 CLS : GO SUB 7256: INPUT "<1> a files- or <
2> z files?";a: INPUT "Input Start File number t
o Merge";m: LET m$=("a"+STR$ m AND a<2)+("z"+STR
$ m AND a>1): DIM o$(1): LET m$=m$+".A$": DIM v$
(1): RANDOMIZE USR 100: LOAD m$ DATA v$(): GO SU
B 7162: DIM o$(96,L2,L3): LET xx=1: FOR n=oa TO
96 STEP 24: DIM v$(24,L2,L3): FOR y=m TO m+3: GO
 SUB 7284: FOR g=1 TO 24: FOR i=1 TO L2: LET o$(
xx,i)=v$(g,i): NEXT i: LET xx=xx+1: NEXT g: LET
m=m+1: NEXT y: LET m=m+1: NEXT n: DIM v$(1): BEE
P 1,ok: CLS : PRINT "Files ";("a" AND a<2)+("z"
AND a>1);" ";m-3;" to ";m;" are combined and rea
dy""to Sort.  ENTER": PAUSE 0: GO SUB 7232: GO
TO 7020
7284 DIM v$(1): LET m$=("sa"+STR$ m AND az<2)+("
za"+STR$ m AND az>1): LET m$=m$+".A$": RANDOMIZE
 USR 100: LOAD m$ DATA v$(): RETURN
7290 REM  reverse groups in line (so) for sort b
y first chr group
7294 CLS : PRINT "Reversing groups in Line ";so:
 FOR n=1 TO L5: LET b$=o$(n,so)
7300 IF b$(LEN b$)=" " THEN  LET b$=b$( TO LEN b
$-1): IF LEN b$>5 THEN  GO TO 7300
7304 FOR g=LEN b$ TO 1 STEP -1: IF g=1 THEN  GO
TO 7316
7308 IF b$(g)=" " THEN  LET o$(n,so)=b$(g+1 TO L
EN b$)+" "+b$( TO g-1): GO TO 7316
7312 NEXT g
7316 IF n>=L5 THEN  GO TO 7364
7320 NEXT n
7324 GO TO 7368
7328 REM Return line so to orig data
7332 FOR n=1 TO L5: LET m$=o$(n,so)
7336 IF m$(LEN m$)=" " AND LEN m$>1 THEN  LET m$
=m$( TO LEN m$-1): GO TO 7336
7340 IF LEN m$=5 THEN  NEXT n: GO TO 7352
7344 FOR y=1 TO L2: IF m$(y)=" " THEN  LET o$(n,
so)=m$(y+1 TO )+" "+m$( TO y): GO TO 7352
7348 NEXT y:
7352 IF n>=L5 THEN  GO TO 7396
7356 NEXT n: GO TO 7396
7360 REM *** Sort First Group
7364 PRINT "Sorting, line ";so
7368 DIM v$(L2,L3): LET n=L5: LET i=n-1: FOR j=1
 TO i: LET k=j+1: FOR L=n TO k STEP -1
7372 IF o$(L,so)(1 TO 4)="    " THEN  GO TO 7384

7376 IF o$(L,so)>o$(j,so) THEN  GO TO 7384
7380 IF o$(L,so)(1 TO 4)<>"    " THEN  LET b$=o$
(L,so): FOR x=1 TO L2: LET v$(x)=o$(L,x): NEXT x
: FOR x=1 TO L2: LET o$(L,x)=o$(j,x): NEXT x: FO
R x=1 TO L2: LET o$(j,x)=v$(x): NEXT x
7384 NEXT L
7388 NEXT j
7392 IF sol=2 THEN  GO TO 7332
7396 CLS : PRINT AT 10,4;"Sort Complete": BEEP 1
,18: PAUSE VAL "200": GO TO 7020
7400 CLS : PRINT AT oj,oo;"Input Key Word": INPU
T y$: INPUT "<1> View or <2> Print out";z: IF z=
2 THEN  INPUT "Tab?";tb: INPUT "nbr lines per pa
ge?";ln: INPUT "Page Nbr?";pg
7404 LET nn=0: FOR n=1 TO L5: IF o$(n,7)( TO LEN
 y$)=y$ THEN  GO SUB 7412
7408 NEXT n
7412 IF z=1 THEN  PRINT "Record# ";n
7416 IF z=2 THEN  LPRINT "Record# ";n: LET nn=nn
+1
7420 FOR y=1 TO L2: IF z=1 THEN  PRINT o$(n,y)
```

requires that a file loaded from disk be accomplished via MENU SELECTION. If a file should be loaded from disk via a direct command, then the "L1 through L5" vars would not be correct and the system management would break down during some selected functions.

The Dbase-1 program does not use a "vars table". The program variables are assigned as needed. When the program is keyed in and has been tested, you can gain around 1000 bytes of FREE by systematically editing all lines and enclosing all numbers in Quotes and preceeding them with VAL. Various "variable Switches" are used for "turning ON or Off" certain functions as menu options are elected. Two examples can be seen in the printing routine at line 7480, where variable "pgc" is used to switch ON or OFF "page control" and variable "ret" is used to switch ON or OFF dual return labels while printing address labels. You can discover other examples of "variable switches" in the LET statements throughout the program after prompts. Some prompts use INPUT for a variable to be assigned <INPUT A>, and others use INKEY$, as <PAUSE 0:LET A=CODE INKEY$-48>. The latter assigns a number key value to the LET variable A.

DIGGING IN TO CHARACTER ARRAY MANAGEMENT: CONCATENATION: Working with Numbers: Thats what Sinclair Computers do best. Learn this and most other things become more understandable. Every character that you see on screen is a "number"! A=65 and a=97 and space=32. To learn others refer to the user manual "Character Set". The character set is known by another term "ASCII CODES". All of the alphabet, plus math symbols, puncuation characters are assigned a number. The number is in consecutive order from zero (48) through number 9 (57). Then A=65 through Z=90. Lower case characters picks up; a=97 through z=122. The gaps are filled with math symbols and puncuation symbols, so we can SORT TEXT by NUMBER comparisons. If we sort a and A, then A comes first because A=65 and a=97. We can even say that in computer syntax: <IF"A"<"a" THEN PRINT "Aa">. The computer evaluates and find TRUE and prints "Aa".

Now lets SORT a string of data. Do it with your computer. <LET a$="aAa"> <IF A$(1)>A$(2) THEN LET a$=a$(2)+a$(1)+a$(3 TO) > <PRINT a$>. This exercise does two things. It compares the first two characters in a$ "by their number value" and then it CONCATENATES a$, re-arranging its characters. If this stumps you then you haven't started off with a grasp of the fact that in any string of characters each character has its own "Index number" in the string. We started off with a$ having 3 characters. A$(1) was a, a$(2) was A, and a$(3) was also a. If a$ had had 10 characters, as <LET a$="ABCDEFGHIJ", then A$(10) would be "J". Count up the characters and this will come into focus. In dealing with CONCATENATIONS we use the "INDEX NUMBER" in the string. We will use that last example of 10 characters: LET B$=A$(5 TO). Now we have "Sliced" some of A$ off and put the Sliced data into B$. <PRINT B$> produces "EFGHIJ".

So now we have learned how to SORT and how to CONCATENATE in one small paragraph! Really, you have! All you need to do is PRACTICE with the computer and you'll have it down pat. Once you can confidently Concatenate, Slice Strings, and SORT, you can join the head table at any computer programmer's convention. And, you'll know more than some at the table. These three operations are the stumbling blocks to learning computer programming, and they are so simple that everyone should STUDY and PRACTICE. Practice is more valuable than study. Now we will get into the SORTING FUNCTION of this Dbase-1 program. These are the functions which provides the versatility of the software, and each uses the simple methods of number comparison and String Contatination.

First we will study the listing, which is all in

```
7424 IF z=2 THEN  LPRINT o$(n,y): LET nn=nn+1: I
F nn>=Ln THEN  LPRINT '''TAB 40;pg: LET pg=pg+1:
 LET nn=0: OUT 127,12
7428 NEXT y
7432 NEXT n: GO TO 7020
7440 LET spc=0: LET ret=0: LET Ln=54: LET nn=0:
CLS : PRINT #0;AT 0,0;"View or Print out?"'"<1>
View  <2> Print out": PAUSE 0: LET x=CODE INKEY$
-48: IF x>2 THEN  GO TO 7440
7442 IF x=2 THEN  INPUT "Enter STARTING RECORD";
ms: INPUT "ENDING RECORD?";mx: INPUT "Enter Nbr
of Lines to Print      (4 for Labels)";Lp: IF Lp=
4 THEN  INPUT "Return Labels Too?                <
1> Yes or <2> No";ret
7444 IF x=2 THEN  GO TO 7480
7448 PRINT #0;AT 0,0;"<1> Whole Records -or-"'"<2
> Single Lines?            ": PAUSE 0: LET r=CODE
INKEY$-48: IF r=2 THEN  CLS : PRINT #0;AT 0,0;"E
nter record Line Number to view(1 to ";l2;")": P
AUSE 0: LET rL=CODE INKEY$-48: CLS : PRINT #0;AT
 0,0;"To Discontinue, Jab Break"
7454 CLS : PRINT #0;AT 0,0;"To Discontinue, Jab
Break": PAUSE 200: CLS
7460 IF r=1 THEN  FOR n=1 TO L1: PRINT n;: ON ERR
  GO TO 7468: FOR g=1 TO L2: PRINT o$(n,g): NEXT
 g: PRINT : NEXT n: STOP
7464 IF r=2 THEN  FOR n=1 TO L1: ON ERR  GO TO 7
468: PRINT n'o$(n,rL): PRINT : NEXT n
7468 ON ERR  RESET : PRINT !"Enter for Menu": PA
USE 0: CLS : GO TO 6804
7480 CLS : INPUT "TAB?";tb: INPUT "Do you want a
n OUTLINE Format?  <1> Yes or <2> No";tb2: INPUT
 "Spaces between Records?";sp: INPUT "Do you wan
t page control            <1> Yes or <2> No";pgc: IF
 pgc=1 THEN  INPUT "Page #";pg: INPUT "Set Print
er and INPUT # Lines   per Page?";Ln
7482 PRINT AT 8,2;"Your Set-up is:"'"TAB 2;"Prin
t Records ";ms;" to ";mx'TAB 2;"Printing lines 1
to ";lp'TAB 2;"At tab ";tb'TAB 2;"Spaces betwee
n records- ";sp': IF pgc=1 THEN  PRINT "page=";p
g'"Lines per page=";Ln: IF ret=1 THEN  PRINT "Re
turn Labels will print"''
7484 IF tb2=1 THEN  PRINT "Outline Format": INPU
T "Correct?  <1> Yes or <2> No";z: LET tb2=(tb=3
 AND tb2<2)+(tb=3 AND tb2=2): IF z=2 THEN  GO TO
 7440
7488 FOR n=ms TO mx: FOR y=1 TO Lp: LPRINT TAB (
tb AND n<2 OR tb2>1)+(tb2 AND n>1 AND tb2<2);o$(
n,y): GO SUB 7510: NEXT y: GO SUB 7516
7490 IF spc=1 AND nn+Lp>Ln THEN  GO SUB 7500
7494 NEXT n
7496 GO TO 6804
7500 LPRINT '''TAB 50;pg: OUT 127,12: LET pg=pg+
1: LET nn=0: PRINT "Service Printer, then ENTER"
: PAUSE 0: RETURN
7510 IF pgc=1 THEN  LET nn=nn+1
7512 RETURN
7516 FOR g=1 TO sp: LPRINT : NEXT g: IF pgc=1 TH
EN  LET nn=nn+sp: IF nn>Ln+sp THEN  GO SUB 7520
7518 RETURN
7520 LPRINT '''TAB 50;pg: OUT 127,12: LPRINT : PR
INT AT 10,2;"Servce printer, Then ENTER": PAUSE
0: CLS : RETURN
7530 CLS : INPUT "<1> Letterhead or <2> Sign Off
?";ls: IF ls=1 THEN  INPUT "<1> Single or <2> Mu
ltiple?";sm
7532 IF ls=1 AND sm=2 THEN  FOR y=1 TO mx
7534 IF ls=1 THEN  FOR n=1 TO 3: LPRINT TAB 28;n
$(n): NEXT n: OUT 127,12: IF sm=2 THEN  NEXT y
7536 IF ls=2 THEN  LPRINT '''TAB 30;n$(5)'''TAB
30;n$(4): OUT 127,12
7538 GO TO 6804
```

* WANTED *
Z88 Programming Articles
By Update Magazine
For the Users

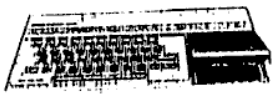
simple basic. This is quite a plate full, and it may take a week or two to key it all in and check the operation. You can tackle it with confidence knowing that this software DOES WORK. The job of Keying it in will accomplish two worthy goals. First, a working understanding of how the program works will be gained. Then you will have a very useful software to use forever. As one keys in projects such as this long program, a full understanding is not usually gained at first. However, there is something about keying in programs that sticks in one's memory and brings later comprehension. The act of doing it is the best way of learning computer programming.

We will start with the "Sort Selection Menu" at line # 7054. Here we see the "L" variables previously discussed. L1=first dimension of the character array O$(x,x,x). L2=second dimension, which is the Number of Lines in the records. The menu asks for you to decide "which line to sort". If you elect to sort LINE #1" then vars "so=1". The next prompt asks for you to elect to sort "by first group of line 1, or last group of line 1". For understanding I will give you a typical "line #1". If Line 1 of the first record were "Mr. Benjamin K. Franklin", and you selected "sort by line 1 and Last Group", then you have set up to sort by "Last Name" of a mail file. You could have elected "first group" and the sort would sort "Mr.", which would serve no useful purpose.

But if you were sorting an "Inventory file" then the first group of line 1 would likely be a "Stock Number", and "sort by first group" would give you sort by Stock Number. The number of Sorting Options vary by the number of LINES in a record. If your data base has 6 lines then you have 12 choices of sorting. If the record contains 20 lines, then there are 40 choices of sorting. The Sorting method is "by first or last distinctive WORD or GROUP of the selected line". During the Sorting, all lines of the record will be replaced as a sorting action occurs. A tremendous number of actions occurs during sorting of a 96 record file. The very least number would be 9216. Each "record sorted" would be 14 more actions by the nested counters. There could be as many as 129024 individual sorting actions for a 96 record file in the few lines beginning at line #7368 and ending at line #7388.

Now lets shift our attention to the actual SORTING routine which begins at line # 7364. Note the REM at line 7360, which states "Sort First Group"! Something is amiss here! We selected "sort by last group of line 1 (Franklin)". Yep! If this line were to operate now, the sort would line up by "Mr.", which would be of no value. So, lets go back to the menu at line # 7054. There we see the next program line (7058), and its statement "IF so<>2 THEN GO TO 7294". Then line number 7294 is where we must continue the examination. At line 7294 the "last group" in the selected sort line is moved ahead to become the "first group" in the line. We do that with the "n counter" to count up all records in the data base. Then each first line of each record is re-arranged so that the character group that we want to sort is placed FIRST in the line (Franklin).

When finished with the reversing procedure, line number 7324 operates to <GO TO line 7368>. Now we can look at line 7368 where the SORTING occurs. At line #7368, we establish a buffer for holding each record as the sorting is accomplished. V$ is dimensioned "V$(L2,L3)" which matches the record size of the data file being sorted. The routine that follows counts up to the END record in the file. With each count, the "first character group" is compared with the first character group of the "last record" in the file. This is a "numeric evaluation" similar to how we compared "a" and "A" in the earlier discussion. Each time the record at the END of the file is SMALLER, then the buffer (V$)



Line Changes for Olliger DOS

```
7152 GO SUB 7256: INPUT "Zip File# ?";z: LET z$=
"z"+STR$ z: SAVE z$ DATA o$(): GO SUB 7232: GO T
O 7020
7232 CLS : INPUT "Re-Install Program Disk, ENTER
  DRIVE # ?";d: LET /d=d: RETURN
7244 LET xx=1: FOR n=1 TO 96 STEP 24: DIM v$(24,
12,13): LET m$=("a"+STR$ s4 AND a<2)+("z"+STR$ s
4 AND a>1): FOR y=1 TO 24: FOR g=1 TO 12: LET v$
(y,g)=o$(xx,g): NEXT g: LET xx=xx+1: NEXT y: SAV
E /m$ DATA v$(): LET s4=s4+1: NEXT n: GO SUB 723
2: CAT : PRINT #0;AT 0,0;"Note the Saved Files,
ENTER": PAUSE 0: GO TO 7020
7256 INPUT "Install Data Disk, then ENTER  Disk
 #";d: LET /d=d: CAT : RETURN
7264 CLS : GO SUB 7256: INPUT "<1> Alpha Save, o
r <2> Zip Save ";az: INPUT "Enter File Nbr?";a:
LET m$=("A"+STR$ a AND az<2)+("Z"+STR$ a AND az>
1): SAVE /m$ DATA o$(): GO TO 7020
7272 CLS : GO SUB 7256: INPUT "<1> Alpha File or
 <2> Zip File";a: INPUT "File Nbr to Load?";m$:
LET m$=("A"+m$ AND a<2)+("Z"+m$ AND a>1): DIM o$
(1,1,1): DIM v$(1,1): LOAD /m$ DATA o$(): GO SUB
 7232: GO SUB 7162: GO TO 7020
7280 CLS : GO SUB 7256: INPUT "<1> Alpha File or
 <2> Zip File";a: INPUT "Start File to Merge?";m
: LET m$=("a"+STR$ m AND a<2)+("z"+STR$ m AND a>
1): DIM o$(1): DIM v$(1): LOAD /m$ DATA v$(): RE
M Continue Same Programming as in line 7280 abov
e.
7284 DIM v$(1): LET m$=("sa"+STR$ m AND az<2)+("
za"+STR$ m AND az>1): LOAD /m$ DATA v$(): RETURN
```

Line changes for the Aerco FD-68:

There are two functions that cannot be done with the Aerco FD-68 because of the lack of a .CHR Extension. These are the "Split and SAVE" and "Load and Combine Small Files". The first change will be to delete those functions from menu.

These adaptations have not been tested and may require some string Concat adjustments.

```
7020> BORDER 0: INK 7: PAPER 0: CLS : PRINT AT 2
,0;"File Management Options"'''1> Remove Repeats
"'"2> Remove Skips"'''3> Move Zip to line 4"'"4>
 Sort Data Base"'"5> Save a Sorted File"'"6> Lo
ad a File"'''7> CLEAR Memory"'"8> Present Key Re
cords"'"   (Line 7 Key)"'''9> View or Print Reco
rds"'"0> To Function Menu": INPUT "Input Choice:
";z$: LET z=CODE z$-48: IF z>9 THEN GO TO 7020
7024 IF z>5 THEN GO TO 7050
7050 GO TO (z=1)*7110+(z=2)*7132+(z=3)*7070+(z=4
)*7054+(z=5)*7264+(z=6)*7272+(z=7)*7126+(z=8)*74
40+(z=9)*7400+(z=0)*6804
7154 GO SUB 7256: INPUT "Zip File # ?";z: LET z$
="z"+STR$ z: MOVE "z$.VAR",: GO SUB 7232: GO TO
7020
7232 CLS : INPUT "Re-Install Program Disk, ENTER
  Which Drive?";a$: RETURN
7240 REM DELETE this line.
7244 REM Delete this line
7256 REM remove rem + this ** CLS : INPUT "Insta
ll Data Disk, Then ENTER   Which Disk?";a$: CAT
"a$";:",: RETURN
7264 CLS : GO SUB 7256: INPUT "<1> Alpha Save- o
r <2> Zip Save";az: INPUT "Enter file Nbr:";a: L
ET m$=("A"+STR$ a AND az<2)+("Z"+STR$ a AND az>1
): LET m$=m$+".VAR": MOVE "a$:m$",: GO TO 7020
7272 CLS : GO SUB 7256: INPUT "<1> Alpha File- o
r <2> Zip file";a: INPUT "File # to Load?";m$: L
ET m$=("A"+m$+".VAR" AND a<2)+("Z"+m$+".VAR" AND
 a>1): DIM v$(1,1): CAT "a$:m$",: GO SUB 7232: G
O SUB 7032: GO TO 7020
7280 REM DELETE this line
7284 DIM v$(1): LET m$=("sa"+STR$ m AND az<2)+("
za"+STR$ m AND az>1): LET m$=m$+".VAR": CAT "a$:
m$",: RETURN
```

receives all record lines of the first record. Then the first record is REPLACED with the LAST RECORD. This leaves two identical records.

Then the LAST record is replaced with the record that is in the buffer (V$). The results from this is that the entire file of records is sorted. Next, at line # 7392 there is a branch. Remember that we had to REVERSE the groups in line 1 to sort by "Franklin" or last name. Here "soL=2" and the branch is to line number 7332. At line 7332, the reverse again takes place to move the "first group" in each record's sorting line back to its original position in the line. When finished with that we come back to line 7396 to present the "sort complete" message and return to menu. Of course, this double reversal is not used when the sorting selection is "First Group".

REPORT CODES: While operating a Basic Program, and mistakes are made, the TS-2068 stops and gives a friendly REPORT CODE at screen bottom. Examination of the Report most times tells one what mistake was made. The suppression of these report codes can be done with programming, but often the program is extended to such length that the DATA CAPACITY is severely limited. But without suppression, the user is left with a STOPPED program with every mistake. The solution is to know the "GO TO" to get back into operation without loss of data. <GO TO 6800> is to the Master Menu of D.Base-1 and is the recovery for all Report Codes. But one should first "THINK, What Caused The Report Code", so that the same mistake is not repeated.

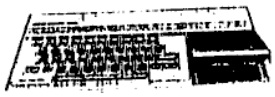
```
File Management Options

1> Remove Repeats
2> Remove Skips
3> Move Zip to line 4

4> Sort Data Base
5> Break Lg File to 4 Files
   and Save
6> Combine 4 Small Disk Files
   to Sort

7> Save a Sorted File
8> Load a large File
9> Clear Memory

0> Present Key Records
   (Line 7 Key)
:> View or Print Records
;> To Function Menu

Input Choice
```

```
        Dbase-1 Menu

   <1> Create File
   <2> Add Records
   <3> View or Print Records

   <4> Correct Records
   <5> SAVE File
   <6> LOAD File

   <7> Sort and Manage
   <8> To Function Menu
   <9> Letterhead or Sign Off

    Touch Selection
```

SINCUS EXCHANGE DISKS AVAILABLE

SINCUS, the Sinclair Users Society of Upstate New York, is offering four PD disks for sale or exchange. These disks are available on the Olliger format (and Larken very soon, we hope), double-sided double density (DSDD), 40 tracks. Disk #101 contains entertainment programs. Disk #102 has business applications. Disks #103 and #104 contain a potpourri of programs. There are over 100 files in all. These disks are being offered at cost for $2.50 each (about 10 cents a program!) The programs are taken from SINCUS members, exchange tapes from other clubs, newsletter articles, CompuServe, and other bulletin boards. For a complete list of titles, send a SASE to the address below. To order send a check or MO ($2.50 for each disk) and indicate which disk(s) you wish to obtain.

John Colonna, SINCUS
28 Guilfoyle Avenue
Binghamton, N.Y. 13903



The "Mitchell VC" Issue Disk for LKDOS
(Oliger DOS Version avail August 1989)

VU-CALC TUTORIAL.
WIDE PRINTER ENHANCEMENT
AND ADDITIONAL NOTES ON VU-CALC.

By Bob Mitchell Willowdale Ontario

==========

Printing a spreadsheet on the tiny 2040 printer does not provide a very useful output. You end up cutting and pasting to get a full display on one piece of paper. Then, to keep the copy from fading or turning brown, it must be photocopied. Hardly state of the art!

Thus provoked, I wrote a revised version of the vu-calc BASIC program <vcalc.B3> to take advantage of my wide printer (Smith-Corona Fastext 80) and the Larken printer interface code (pif) contained in the Version 3 Eprom available from Larken Electronics.

Next came a BASIC routine to perform the spreadsheet printout <vcalcP.B2>; this will work fine, but the printout is very slow. The next step was to compile the printout routine via Timachine and the result was much faster, quite tolerable in fact.

==========

Let's take <vcalc.B3> first.

Enter vcalc.B3 (listing 1) and SAVE by going to line 9900 (or 9910 for a Larken AUTOSTART SAVE).

To use vcalc.B3 requires the vu-calc machine code that appeared on the original tape. If you have this on disk, skip the next paragraph.

If you are going to load the code "vc" from the original Psion cassette, LOAD vcalc.B3, BREAK into BASIC, position the tape at the start of the code and use LOAD "" CODE. When the code is loaded, enter <GO TO 12> to carry on. Now is a good time to SAVE this code to disk. Use <PRINT #4: SAVE "v.C1"CODE 29327,5230.

There is a display of the number of formula slots used out of the 100 available; this helps keep track of how many have been used. Careful assignment and editing of formulae is essential to avoid running out of slots, especially in a complex spreadsheet.

Two HELP files available on the Update disk may be displayed on screen using <taswide>. Alternatively, you can get a copy on your wide printer by using the Larken pif. But you must have the Version 3 Larken Eprom to do this. Use the following procedure to get this print out:

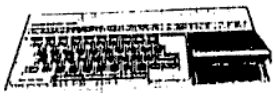
1. PRINT #4: POKE 16090,63: REM line length
2. PRINT #4: POKE 16092,10: REM line feed
3. PRINT #4: POKE 16094,8: REM left margin
4. PRINT #4: LPRINT "vcC?.CT" (or vcT?.CT).

The two Help Files are:

1. Commands. <vcC?.CT> This is essentially the detail included in the Vu-Calc manual, with a few additional notes.
2. Tutorial. <vcT?.CT> This is the file you are viewing now. It not only explains how the printing takes place on the wide printer but also includes additional information on vcalc.B3.

LISTING 1

```
 10 ON ERR  RESET : CLEAR VAL "
29327": CLS : RANDOMIZE USR VAL
"100": OPEN #VAL "4","dd"
 11 PRINT #VAL "4": LOAD "v.C1"
CODE
 12 PRINT #VAL "4": OPEN #VAL "
3","lp": PRINT #VAL "4": POKE VA
L "16090",VAL "127": PRINT #VAL
"4": POKE VAL "16094",VAL "8": P
RINT #VAL "4": POKE VAL "16092",
NOT PI
 20 CLEAR : GO TO VAL "3200"
 900 LET lenc=PEEK bfre+VAL "256
"*PEEK (bfre+1)-zz: RETURN
1000 GO SUB VAL "1100": PRINT #V
AL "4": LOAD a$CODE : CLS : GO T
O USR e2
1100 GO SUB VAL "1200": PRINT #V
AL "4": CAT ".CV",: INPUT "name?
 (eg,vinc)" LINE a$: LET a$=a$+
".CV": RETURN
1200 PAPER VAL "7": INK VAL "0":
 BORDER VAL "1": CLS : RETURN
2000 IF PEEK VAL "63000"<>VAL "2
05" THEN  PRINT #VAL "4": LOAD "
v2.CP"CODE VAL "63000"
2005 RANDOMIZE USR VAL "63000"
2010 GO TO USR e3
3000 GO SUB VAL "1200": PRINT "1
>EXIT"''"2>CLEAR WS"''"3>RESUME"
''"4>DATA via TAPE"''"5>HELP!"''
"Formulae used= ";PEEK VAL "3456
6":  INPUT "OPTION? ";a
3010 IF a>NOT PI AND a<VAL "6" T
HEN  GO TO (VAL "3000"+a*VAL "10
0")
3020 GO TO VAL "3e3"
3100 GO SUB VAL "1200": STOP
3200 CLEAR : DIM b$(VAL "100"):
DIM c$(VAL "20"): GO SUB VAL "12
00"
3201 LET e1=VAL "29764": LET e2=
e1+VAL "3": LET e3=e2+VAL "3": L
ET zz=e1+VAL "4789": LET bfre=e1
+VAL "4798": LET dcontx=e1+VAL "
4376": LET xcol=e1+VAL "3116": L
ET xrow=xcol+LN PI: GO TO USR e1

3300 CLS : GO TO USR e2
3400  INPUT "1=save  0=load ";s1
: IF NOT s1 THEN  CLS : LOAD ""C
ODE : GO TO USR e2
3410 IF s1 THEN  GO SUB VAL "900
": SAVE a$CODE zz,lenc: CLS : GO
 TO USR e2
3500 INPUT "1=commands 0=tutoria
l ";cc: IF NOT cc THEN  CLS : LE
```



Also included is an option in the vu-calc BASIC program to save or load data files via tape.

==========

Enter vcalcP.B2, the wide printer routine (listing 2) and save it.

Several lines send commands to the printer: Line 2008 selects PICA font and turns CONDENSED mode OFF. Line 2010 sets CONDENSED mode to ON; this is a fairly standard command <OUT 127,15>. Lines 2075 and 2120 do a Line Feed while line 2135 does a form feed. Check your printer manual for correct commands and change these lines accordingly.

You may want to check the BASIC before compiling. Activate your LKDOS printer interface code, load a data file and GO TO 110. Your printer must be turned on. The spreadsheet will be printed, albeit very slowly. Make sure the BASIC is working to your liking before compiling it.

Clear the computer and LOAD Timachine. LOAD vcalcP.B2 into Timachine (use RANDOMIZE USR 100: LOAD "vcalcP.B2) and do the compilation. SAVE the compiled code to disk and clear the computer. LOAD the compiled code back into the computer and SAVE it again as v2.CP CODE 63000,length. (length is given in the Timachine output.)

Make sure your wide printer is ON.

LOAD vcalc.B3 and using vu-calc option <#l>, (that's el for Lima), load any available spreadsheet data or make up one. There is a sample on the Update disk (vgh2). Use the vu-calc command <#p> to start the wide printer routine.

You are prompted for some inputs. It will be necessary to determine the dimensions of the wide printout as you view the screen and before going to the print routine.

1. extra space between columns? y/n. Entering <y> will make reading columns with seven numbers easier. This will sometimes break up words that span two or more columns. Unless the numbers are large enough to fill up the column width, <n> will usually suffice.

2. No. of cols. Up to 9 columns (spaced) or 10 (unspaced) will result in a PICA sized printout; otherwise CONDENSED mode will be switched in. If too many columns are called for, an error report will appear and this prompt will be repeated.

3. start at col? (1-50). Enter the column number shown at the top of the vu-calc grid.

4. start at row? (a TO ax). Enter the row letter shown at the left side of the vu-calc grid.

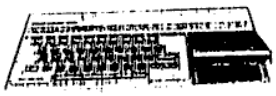
5. end at row? (b TO ax). Enter the row letter at which the printout is to stop. The <b> in this example will change to correspond to the next letter after the start row letter.

6. paper eject? Enter 1 for yes or 0 for no.

```
T i$="vcT?.CT": GO SUB VAL "3600
": GO TO VAL "3000"
3510 CLS : LET i$="vcC?.CT": GO
SUB VAL "3600": GO TO VAL "3000"

3600 IF PEEK VAL "64300"<>VAL "2
45" THEN  PRINT #VAL "4": LOAD "
taswi.Cx"CODE
3610 ON ERR  GO TO VAL "3620": P
APER VAL "7": INK NOT PI: BORDER
 VAL "7": CLS : RANDOMIZE USR VA
L "64300": PRINT CHR$ VAL "3";:
PRINT #VAL "4": PRINT i$
3620 ON ERR  RESET : PRINT CHR$
VAL "2": PRINT "Press a key": PA
USE NOT PI: RETURN
4000 GO SUB VAL "1100": GO SUB V
AL "900": PRINT #VAL "4": SAVE a
$CODE zz,lenc: CLS : GO TO USR e
2
5000 LET c$=STR$ VAL b$: GO TO U
SR dcontx
9000 GO SUB VAL "1200": PRINT ''
'''ERROR was at''''COLUMN ";PEEK
 xcol+VAL "1"
9100 LET a=PEEK xrow: LET b=INT
(a/VAL "26"): LET a=a-b*VAL "26"
: LET d$=CHR$ (b+VAL "64")+CHR$
(a+VAL "65"): IF b=NOT PI THEN
LET d$=d$(2)
9200 PRINT "ROW    ";d$: INPUT "
Press ENTER to continue";b$: GO
TO USR e2
9500 INPUT a: PRINT PEEK a+VAL "
256"*PEEK (a+SGN PI): GO TO VAL
"9500"
9900 CLEAR VAL "29327": RANDOMIZ
E USR VAL "100": SAVE "vcalc.B3"
 LINE VAL "10": STOP
9910 CLEAR VAL "29327": RANDOMIZ
E USR VAL "102": GO TO VAL "10"
```

LISTING 2

```
 50 REM !USR 63000
 60 REM ! LPRINT
 70 REM !INT +xs,le,end,x,y,r,h
,eject,z,q,f,g
 80 REM !LEN $<=2
 90 REM ! LIST
100 REM ! OPEN #
110 GO TO 2000
1000 IF IN 127<>237 THEN  GO TO
1000
1010 RETURN
2000 CLS : INPUT "extra space be
tween columns? y/n"; LINE s$: IF
 s$<>"y" AND s$<>"n" THEN  GO TO
 2000
```



VU-CALC INSTRUCTIONS

COMMAND PANEL. Four types of inputs:

1. TEXT. Enter column or row headings or subheadings, names, or any other labels you choose.
   First, position the red cursor to the box on the 50 x 50 grid where you wish to start your text.
   Next, Enter< " > and up to 28 characters. Press ENTER to transfer text to the grid.

2. DATA. Position cursor type data and press ENTER. Numerical data may be entered directly.

3. FORMULAE. Enter these directly. Position cursor to box which will print the result of the formula. The answer will appear in the chosen box with the formula appearing just under the box and to its left. Returning to this box will display the formula.
   .Syntax of a formula: constants (numbers), references to numbers in other boxes and simple arithmetical operators +, -, /, and *. Refer to other boxes by their coordinates, rc where < r > is the row letter(s) and < c > is the column number.
   eg. B1*1.03
   D7-C7
   D12*(B2+1.5)/C1

   .A formula can be applied to rows or columns of numbers by using the Repeat Command (see below).
   .Sum facility enables you to add a row or column or part thereof or a rectangular block of boxes.
   Use the form &f:l where f=top left hand box
   and l=bottom right hand box.
   Note box f must always be above and/or the the left of the box reference.
   eg. &A2:B4 (sums boxes A2+A3+A4+B2+B3+B4)
   &C3:C10 (sums row C3 to C10)
   &B5:D5 (sums column B5 fo D5)

4. COMMANDS. Press < # > and the two line command panel appears.
   #B -- Blank: to delete contents of current box
   ready for updated data .
   #C -- Calculate: Recalculates entire table after entering new or changed data, etc.
   #E -- Edit: Allows formula to be changed. If this is not done faithfully, formula capacity will be needlessly wasted.
   NB. There is no provision for deleting a formula and thereby restoring the capacity by one.
   #F,c,f,j -- Format, Format, column, format-type, justified: Determines how a number will be shown in a given col. Defined by three parameters -- c,f, and j
   c = one or two digit number) or (< A > ie, all
   specifying a column) (for entire table
   f = format type required i = integer
   $ = two decimal places
   g = general
   j = justified l = left
   r = right
   #G,rc -- GO TO box designated by row r and col c.
   #L -- Load data from disk. Type in file name with no extension. (extension is always < .SV >.

```
2002 LET xs=7
2003 IF s$="y" THEN  LET xs=8
2004 LET end=PI-PI: INPUT "No. o
f cols ";x: LET x=x*xs
2008 GO SUB 1000: OUT 127,27: GO
 SUB 1000: OUT 127,80: GO SUB 10
00: OUT 127,18
2009 IF x>120 THEN  PRINT "too m
any columns": PAUSE 60: GO TO 20
00
2010 IF x>75 THEN  OUT 127,15
2020 INPUT "start at col? (1 TO
50)";y: LET y=y-SGN PI
2030 INPUT "start at row? (a TO
ax)";z$
2035 IF LEN z$=SGN PI THEN  LET
z$=" "+z$
2040 IF z$(2)="" THEN  LET h=COD
E z$(1)-97
2050 IF z$(2)<>"" THEN  LET h=(C
ODE z$(2)-97)+(26 AND z$(1)="a")

2052 IF end THEN  LET r=h: RETUR
N
2055 LET z=h
2060 INPUT "End at row? (";CHR$
(h+98);" TO ax)";z$: LET end=SGN
 PI: GO SUB 2035
2063 INPUT "paper eject? 1=yes 0
=no ";eject
2065 LET r=r-z: IF r<=PI-PI THEN
  GO TO 2060
2070 LET q=34573+(7*y)+(350*z)
2072 IF xs=8 THEN  LET le=x-1-(x
/8)
2074 IF xs=7 THEN  LET le=x-1
2075 LPRINT CHR$ 10
2080 FOR f=q TO q+(r*350) STEP 3
50
2090 FOR g=0 TO le
2100 LPRINT CHR$ PEEK (f+g);
2105 IF xs=8 AND (g+1)/7=INT ((g
+1)/7) THEN  LPRINT " ";
2110 NEXT g
2120 LPRINT CHR$ 10
2130 NEXT f
2135 IF eject THEN  OUT 127,12
2137 LPRINT
2140 RETURN
```

WANTED-- Your Contribution

Most articles and programs in UPDATE are contributed by the Subscribers. We Sinclair Users support each other like no other user group. We survive upon INWARD CONTRIBUTING SUPPORT. If you have a good program or utility, let Update Magazine give it to others who need it. We dont care if you dangle a particutical or two in your prose, and we aren't particular about the format. Its the programming that counts! -and the participation.

#P -- Print a copy to the wide printer.
#Q -- Quit and 1. exit the program.)
2. clear the worksheet.)
3. resume vu-calc operation.) press ENTER
4. save/load data via tape.)
5. view help files.)

#R,rc,f:l -- Repeat contents of box rc over range of boxes which may be a row or column or part thereof or a rectangular block of boxes. It means: "repeat contents of box rc in all boxes from <f>irst to <l>ast"
Similar to Sum Facility above.
Also repeats a formula for box rc across a row or down a column but not over a block of boxes.
eg, if box A2 has formula < 1.03*A1 > then #R,A2,A3:A10 will automatically increase the value of A in the formula by 1 for each succeeding box.
Thus box A3 would have formula < 1.03*A2 >;
box A4 would have formula < 1.03*A3 >; etc
For columns, the letter would increase -- B to C, C fo D, etc, as the formula moved down the column.
Prefix a box reference with < $ > to render it an absolute box which does not change when repeated.

In the above example, the formula 1.03*$A1, when applied to any box in row A would remain the same.

#S -- Save file to disk, without specifying the extension, which will be < .CV >. Upon completion of the save, the program with all data intact will return to the screen.

#T,r(or c),r(or c) -- Transfer a row or column defined in the first parameter to the row or column defined by the second parameter. A row cannot be copied to a column or vice versa.

5. If a formula is entered which references a blank box or a box with text rather than data, the program will stop with an error report. GO TO 9000 and press < ENTER >. The display will return to the table and you may continue.

Editors Note: The "Mitchell VC Disk" contains: <1> Operating Listing 1, <2> Operating Compiled Listing 2, <3> Un-compiled Listing 2 for user adaptations, <3> Supporting Code, <4> A Saved sample View Calc Spreadsheet for an example to work with. Update advises all to obtain a View Calc Cassette from your dealer to have the original back up program and the manual. You can likely buy it for less than $16.00. The disk update of this software makes it a very nice addition to anyone's computing capability.

Dear Ann Landers

I'm writing to tell you my problem. It seems I have been married to a sex maniac for the past 17 years.

He makes love to me regardless of what I'm doing; ironing, washing dishes, sweeping, etc. I would like to know if there is anything to prevent this? Please print this letter in your column. Everything you print is true. On second thought just forget it. Susan

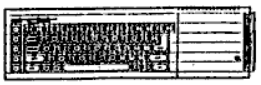
SMUG (Sinclair Milwaukee User Group) is a Sinclair user group that has been active since early 1983. We currently have a membership of 30 active Sinclairites. We meet once a month all year long on the 1st Wednesday of the month. We have 4 SIGS, games, hardware, Spectrum/QL and "C". The SIGS meet at various times of the month and are a great way to gain information on a specific subject. The club sponsers educational classes on varying subjects. Our address is SMUG P.O. Box 101 Butler WI 53007.

We have all models of the Sinclair family, except for the Z88, in the club but our main thrust is the TS2068 and the QL. Our disk systems consist of the AERCO and the Oliger. Our printers are varied we have 80 col. to 135 col. we even have a laser printer in the group, although it is not on a Sinclair computer. There are a few owners of the Westridge 2050 modem with some faster modems on the QL.

We publish a monthly newsletter called SMUG Bytes. The subscription rate is $10 per year. We also exchange the newsletter with over 20 groups throughout the US and Canada.

Source for 3" Diskettes for Amdisk III

3" Diskettes: 10 for $40.00+ $2.50 postage. Check or Money order to: Herb Moulds, PO Box 312, Lathrop Village, MI 48076. (313 533 8797)



DEC. 17, 1988

CHARLES E. REISE
7614 LOVELLA
ST. LOUIS, MO

BILL JONES
UPDATE MAGAZINE
1317 STRATFORD AVE
PANAMA CITY, FL. 32404

DEAR BILL,

ENCLOSED ARE INSTRUCTIONS FOR SETTING UP CHECK BALANCING USING QL ABACUS PUGRAM. ALSO SOME GENERAL INSTRUCTIONS FOR ENTERING WHAT WHERE WHY AND HOW. HOPE THEY ARE WRITTEN IN AN UNDERSTANDABLE MANNER.

SEASONS BREETINGS,
CHARLES E. REISE

P. S. MAY YOUR CHECKBOOK BALANCE FOREVER.

SETTING UP ABACUS FOR MONTHLY CHECKING ACCOUNT

INSERT SS IN MDV1. PRESS F1 TO LOAD. THIS IS, IF YOU ARE USING A A MONITOR. THE LARGE AREA IN THE CENTER IS THE GRID. COLUMNS ARE LETTERED. ROWS ARE NUMBERED. THE CURSUR IS IN CELL A1. CLOUMNS ARE 10 CHARACTERS WIDE. ONE CHARACTER SPACE BETWEEN EACH COLUMN. ONLY SEVEN. ARE SHOWING(THERE ARE 64)AND THEY MAY BE VARIED IN WIDTH, FROM 1 TO 66 CHARACTERS. THERE ARE 255 ROWS. DIRECTIONS FOR SETTING UP FOLLOW

PRESS F3 G W 8 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 9 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 6 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 8 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 9 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 9 ENTER ENTER ENTER RIGHT ARROW
PRESS F3 G W 9 ENTER ENTER ENTER RIGHT ARROW

LEAVE CELL H1 AS IS. YOU NOW HAVE EIGHT COLUMNS. PRESS F5 B1 ENTER. CURSOR IS NOW IN CELL B1. CELL A1 WILL BE FOR DATE, WHICH MUST BE ENTERED AS TEXT(SHIFT QUOTATION)LATER. "12/25/88. ALL TEXT ENTRIES MUST BE ENTERED IN THIS MANNER.

YOU ARE IN B1. TYPE IN AS TEXT"CK_NUM ENTER. GO TO D1(RIGHT ARROW) TWICE. ENTER IN D1- TO___ -WHICH MEANS TO WHOM CHECK IS WRITTEN. THIS IS JUST A LABEL FOR E1 WHERE THE NAME WILL BE ENTERED. F1 WILL BE BLANK. TYPE IN BAL_FORW IN G1. PUT A ZERO (0) IN H1. NUMBERS AND FORMULAE ENTERED IN A CELL SHOULD NOT BE PRECEEDED BY SHIFT QUOTATION. JUST ENTER THEM. AND PRESS ENTER.

NOW TO ROW 2. PRESS F5 A 2 ENTER. ARE YOU THERE? CELL A2, C2, E2, ARE LABELS FOR B2, D2, AND F2. THEY ARE CALLED DEP_1, DEP_2, AND DEP_3 RESPECTIVELY. THE ACTUAL FIGURES WILL BE ENTERED IN B2, D2, AND F2. G2 WILL BE LABELED INT AND WILL BE ENTERED IN H2. FOR NOW, ENTER LABELS IN THEIR RESPECTIVE CELLS, AND 0 IN B2, D2, F2, AND H2.

CELL A3, (F5 A 3) REMEMBER!! TEXT, TOT_BAL, RIGHT ARROW, 0, RIGHT ARROW, RIGHT ARROW. WHERE ARE YOU? D3. TYPE CK_AMT, GO TO E3, TYPE 0. SKIP F3. IN G3 TYPE FORW_BAL, THEN IN H3, 0.

PRESS F5 A 4. TOT_DEPS. B4 0, C4 BLANK, D4 TOT_CKS, E4 0, F4 # OF CKS, G4 1, H4 **********.

GO TO (F5) A 5. TYPE ROW=REPT("-",WIDTH()+1) ENTER.

GO TO A6, PRESS F3 C ENTER ENTER.

GO TO A1, PRESS F3 J ENTER R ENTER ENTER.

PRESS F3 U ENTER ENTER. CHANGE RANGE TO B1:B9 ENTER. GO TO D1.

PRESS F3 U ENTER ENTER. CHANGE RANGE TO D1:F9 ENTER. GO TO H1.

PRESS F3 U ENTER ENTER. CHANGE RANGE TO H1:H9 ENTER

---CELL FORMULAE---

ENTER THESE FORMULAE IN CORRESPONDING CELLS.

B3---H1+B2+D2+F2+H2
B4---B2+D2+F2+H2
C6---C1+1
B8---H6+B7+D7+F7+H7
B9---B4+B7+D7+F7+H7
H3---B3-E3
E4---E3+0
H6---H3+0
H8---B8-E8
E9---E8+E4
G9---G4+1

GO TO A11. PRESS F3 C. CHANGE RANGE TO A6:A10. ENTER ENTER. THIS WILL COPY CHECK # 2 TO CHECK # 3. MOVE TO A16. COPY CHECK # 3 TO # 4. CONTINUE THIS PROCEDURE UNTIL CHECK # 19 HAS BEEN COPIED. YOU NOW HAVE 20 CHECKS FOR THE MONTH.

PLACE FORMATTED MDV IN MDV2. PRESS F3 S. TYPE JAN ENTER THEN SAVE FEB, F3 S FEB ENTER-F3 S MAR ENTER-F3 S APR ENTER-F3 S ENTER MAY-F3 S JUNE ENTER WILL FILL ONE CARTRIDGE. JUST REMOVE AND INSERT ANOTHER TO CONTINUE THROUGH THE REST OF THE YEAR.

---------- PRINTER SET-UP ----------

By Walt Morais

INTRODUCTION

Many programs have been written for setting up a printer. I find this program useful for printing basic files in the character setting desired with the margin/line format I choose.

I use this program primarily for printing basic files. I use it a lot for printing downloaded files and Easyplex messages from Compuserve. The program uses a different type of menu which you may find interesting. It is a basic version of the "Pull-Down" menu structure. I use the program quite often with QUILL. Use caution, however in setting the margin/line functions so they do not conflict with the same functions in QUILL. With minor program changes, this program can print multiple copies.

PROGRAM OPERATION

When the program is loaded and run, you are presented with a menu of four margin/line choices, followed by six choices of character settings. The top menu item is highlighted.

The up and down cursor keys select a menu item which will be highlighted. To initiate action on this selection, press <ENTER>. Additional instructions will be presented below the menu box.

In the case of margin/line functions, enter the value desired, press <ENTER> and the value entered will be posted opposite the menu item. For the character setting choice, press <ENTER>, and the instruction <F1>-"ON" <F2>-"OFF" is displayed below the menu box. Selecting the desired key will post its function opposite the menu item. (Note: "OFF" is a blank). The up and down cursor keys can be used to change or correct any selection.

After the desired margin/line and character settings are selected, press <F5> to send the commands to the printer. (printer is assumed on). A further message will appear below the menu box stating: "Printer codes have been entered as selected", " Do you wish to print a file ? (y/n)". Selecting "y" will produce a request of: "Enter device " (enter mdv or flp desired. Note the " " after the device is not needed). After entering the device, the request of: " Enter file name " appears. After entering the file name, the file will be printed with the margin/line and character settings selected. When the printing is complete, a message will state: "Printing complete (re-run for further use)".

If "N" is selected when asked if you desire to print a file, the message: "PROGRAM TERMINATED" is displayed, followed in two seconds by the restoration of the program windows to the turn-on default windows and the clearing of the program. The program is not "crash proof". Should this occur, just re-run.

The program listing contains information on the functions of each listing block; however, additional information is presented below:

Line	Function
5 to 80	Program initialization, sets windows, variables, and prints menu.
105 to 180	Provides menu selection by changing highlighted menu item.

205 to 275	Toggle functions ("ON" -"OFF") for character settings.
305 to 365	Printer character commands. Note. These codes are for a QL printer. Change to suit your printer.
405 to 455	Message commands and input requests.
505 to 530	Data (Menu items).
605 to 665	Procedures. (These were used to make the listing less cluttered).

I have used this PRINTER SET-UP program for some time, and I find it quite useful. I would appreciate any comments, either to my CIS number contained in the program, or to my address: 8664 Sturbridge Dr. Cincinnati, OH 45236. Please include a SASE if a reply is requested.

```
5  REMark  ------------------------------------------------
10 REMark ¦              Printer Set-up Program              ¦
15 REMark ¦         By Walt Horais - CIS #71641,664          ¦
20 REMark  ------------------------------------------------
25 REMark  ##########  Initialization  ##########
30 CLEAR: RESTORE : WINDOW 512,202,0,0: PAPER 0: BORDER 1,2: CLS
35 OPEN #3,ser: PRINT #3,CHR$(27);"@": PRINT #3,CHR$(24): CLOSE #3
40 STRIP 0: AT 1,29: INK 6: PRINT "PRINTER FORMAT"
45 DIM a$(10,26): x=1: y=3: p=1: rm=0: lm=0: pg=0: sk=0: e=0
50 i=0: b=0: h=0: pr=0: ds=0: g=0: k=0: l=0: r=0
55 READ a$(1), a$(2), a$(3), a$(4), a$(5)
60 READ a$(6), a$(7), a$(8), a$(9), a$(10): STRIP 6: INK 0
65 AT 3,23: PRINT a$(1): STRIP 4: AT 4,23: PRINT a$(2)
70 AT 5,23: PRINT a$(3): AT 6,23: PRINT a$(4): AT 7,23: PRINT a$(5)
75 AT 8,23: PRINT a$(6): AT 9,23: PRINT a$(7): AT 10,23: PRINT a$(8)
80 AT 11,23: PRINT a$(9): AT 12,23: PRINT a$(10)
105 REMark  ##########  Menu Selection  ##########
110 REPeat loop
115 LET z=CODE(INKEY$(-1)): SELect ON z
120   =208:  IF x=1
125             GO TO 110
130          ELSE : x=x-1: y=y-1: p=p-1: STRIP 6: AT y,23
135             PRINT a$(x): STRIP 4: AT y+1,23: PRINT a$(x+1)
140   =216:  IF x=10
145             STRIP 6: AT y,23: PRINT a$(x)
150          ELSE : x=x+1: y=y+1: p=p+1: STRIP 6: AT y,23
155             PRINT a$(x): STRIP 4: AT y-1,23: PRINT a$(x-1)
160    = 10:  IF p=1: flip: clr: pl: INPUT"Enter Right Margin     ";r:
               IF r>0: rm=1:clr: pu: PRINT r: flop
165          IF p=2: flip: clr: pl: INPUT"Enter Left Margin      ";l:
               IF l>0: lm=1: clr: pu: PRINT l: flop
170          IF p=3: flip: clr: pl: INPUT"Enter Page Length      ";g:
               IF g>0: pg=1: clr: pu: PRINT g: flop
175          IF p=4: flip: clr: pl: INPUT" enter Skip Lines      ";k:
               IF k >0: sk=1: clr: pu: PRINT k: flop
180          IF p=5 OR p=6 OR p=7 OR p=8 OR p=9 OR p=10: flip: pl: info: flop
205 REMark  ##########  Toggle Function  ##########
210   =232:  num: flip: clr: mark: c=1: flop:       REMark <F1>-"ON"
215          IF p=5 AND c=1: e=1
220          IF p=6 AND c=1: i=1
225          IF p=7 AND c=1: b=1
230          IF p=8 AND c=1: h=1
235          IF p=9 AND c=1: pr=1
240          IF p=10 AND c=1: ds=1
245   =236:  num: flip: clr: rub: c=0: flop:       REMark <F2>-"OFF"
```

```
250           IF p=5 AND c=0: e=0
255           IF p=6 AND c=0: i=0
260           IF p=7 AND c=0: b=0
265           IF p=8 AND c=0: h=0
270           IF p=9 AND c=0: pr=0
275           IF p=10 AND c=0: ds=0
305 REMark  ##########  Code Selection  ##########
310   =248:   OPEN #3,ser
315           IF rm=1: PRINT #3, CHR$(27);"Q";CHR$(r);
320           IF lm=1: PRINT #3, CHR$(27);"l";CHR$(l);
325           IF pg=1: PRINT #3, CHR$(27);"C";CHR$(g);
330           IF sk=1: PRINT #3, CHR$(27);"N";CHR$(k);
335           IF e=1:  PRINT #3, CHR$(27);"M";
340           IF i=1:  PRINT #3, CHR$(27);"4";
345           IF b=1:  PRINT #3, CHR$(27);"E";
350           IF h=1:  PRINT #3, CHR$(27);"x1";
355           IF pr=1: PRINT #3, CHR$(27);"p1";
360           IF ds=1: PRINT #3, CHR$(27);"G";
365 CLOSE #3
405 REMark  ##########  Commands  ##########
410 AT 15,15: PRINT " Printer codes have been entered as selected "
415 AT 17,18: PRINT" Do you wish to print a file  ?  (y/n) "                    "
420   =89,121: STRIP 0: AT 17,18: PRINT "                                      "
425            AT 15,15: PRINT "
430 AT 17,23: INK 6: INPUT "Enter Device:      ";dev$
435 AT 18,23: INPUT "Enter file name:   ";fil$
440 COPY_N  dev$&"_"&fil$ TO ser1
445 CLS: AT 9,20: PRINT"Printing complete  (re-run for further use)"
446 BEEP 1500,25: EXIT loop
450   =78,110: CLS: AT 9,20: CSIZE 2,1: PRINT " Program Terminated "
451            BEEP 1500,25: CSIZE 0,0: PAUSE 120
452            WINDOW #2,256,202,0,0: PAPER #2,6: INK #2,2
453            BORDER 1,6,0: CLS: WINDOW 256,202,256,0: PAPER 2
454            INK 6: BORDER 1,6,0: CLS: NEW
455 END IF : END SELect : END REPeat loop
505 REMark  ##########  Data  ##########
510 DATA " Right Margin    (10-80)"," Left Margin     ( 0-70)"
515 DATA " Page Length     (  -66)"," Skip Lines              "
520 DATA " Elite Character       "," Italic Character       "
525 DATA " Bold Character        "," High Quality           "
530 DATA " Proportional          "," Double Strike          "
605 REMark  ##########  Procedures  ##########
610 DEFine PROCedure flip: STRIP 0: INK 6: END DEFine
615 DEFine PROCedure flop: STRIP 4: INK 0: END DEFine
620 DEFine PROCedure clr: pl: PRINT"                                           "
625 END DEFine
630 DEFine PROCedure mark: pu: PRINT'ON': c=1: END DEFine
635 DEFine PROCedure rub: pu: PRINT "  ": c=0: END DEFine
640 DEFine PROCedure pu: AT p+2,53: END DEFine
645 DEFine PROCedure pl: AT 14,23: END DEFine
650 DEFine PROCedure info: PRINT'<F1>-"ON"     <F2>-"OFF" '
655 END DEFine
660 DEFine PROCedure num: IF p=5 OR p=6 OR p=7 OR p=8 OR p=9 OR p=10
665 END DEFine
```

---------- LISTSCAN ----------

By Walt Horais

INTRODUCTION

I am "well over 40" years of age, and often when writing a basic program, I find that my memory is a bit hazy on just how to write a desired program segment. Often, I have written that segment before, and I examine program listings to find it.

The "list" command on the QL is not the easiest to use when examining

a listing for a specific segment. When you see the segment scroll by, if you can note its listing numbers, you can list those numbers to a printer. I desired a more efficient way to examine and print segments of a program listing; this prompted me to write the LISTSCAN program.

I attempt to write programs in segments of 100 line numbers, in 5 digit increments (20 lines will fit nicely on the QL display). This results in a program segment that is easy to display and print. The LISTSCAN program is written in basic, located at line number 30,000. This enables quite lengthy programs to be merged and examined.

PROGRAM OPERATION

Once the LISTSCAN program is loaded and run, a window will be displayed with the request: "Enter device " (enter mdv or flp). Note the "_" after the device is not needed. After entering the device, a request of "Enter file name " is displayed. After entering the file name, the program starts. BE PATIENT ! The program is slow, particularly on long programs. After a program to be examined is merged, the LISTSCAN program determines the lowest line number and the highest line number. This is done to handle different starting line numbers, and stop the program from "running away" after the highest line number.

Finally, the first block of listing is displayed, with a menu stating: "<f>orward <b>ack <p>rint <n>ew file and <q>uit". <f>orward and <b>ack move the displayed listing block by 100 line numbers. <p>rint will print a displayed listing to your printer. <n>ew file is just that; and you are returned to the initial screen request of "Enter device" and "Enter file name". <q>uit restores the program windows to the turn-on default windows and clears the program.

Upon reaching the highest line number display, attempts to continue higher will result in a beep and the message: "Highest display", with a return to the highest number display. The same sequence occurs at the lowest display except the message states: "Lowest display". The program is not "crash proof", but if it does crash while examining a listing, it can be re-run. But before doing so, all listing lines of the program being examined must be deleted. (dline first to last, followed by run works nicely).

COMMENTS

The program listing contains information on the function of each listing block; however, additional information is presented below:

Line	Function
30005 to 30090	Program initialization, sets windows, defines device and file names, determines lowest and highest line numbers.
30105 to 30190	Prints menu, selects higher or lower displays, (<f>orward or <b>ack), sounds alert and prints display limits.
30205 to 30240	Menu commands of: <p>rint, <n>ew file, <q>uit.
30305 to 30330	Procedures (used to make the program less cluttered).

I hope you will find this program useful. I would appreciate any comments, either to my CIS number contained in the program, or to my address: 8604 Sturbridge Dr., Cincinnati, OH 45236. Please include a SASE if a reply is requested.

```
30005 REMark  ------------------------------------------
30010 REMark  |              Listing Scan Program        |
30015 REMark  |        By Walt Horais - CIS # 71641,664  |
30020 REMark  ------------------------------------------
30025 REMark   ##########  Initialization  ##########
30030 DIM dev$(5): DIM fil$(15): WINDOW 512,256,0,0: PAPER 0: CLS
30045 OPEN #4, scr_512x202a0,0: PAPER #4,4: INK #4,0
30050 BORDER #4,1,6: CLS #4
30055 AT 8,20: PAPER 4: INK 0: INPUT,"Enter device:      ";dev$
30060 AT 10,20: INPUT,"Enter file name:   ";fil$
30065 MERGE dev$&"_"&fil$
30070 OPEN #3, dev$&"_"&fil$: INPUT #3, first: f=first
30075 REPeat loop
30080    IF EOF (#3) THEN EXIT loop
30085    INPUT #3, last: l=last
30090 END REPeat loop: CLOSE #3
30105 REMark  ##########  Display List  ##########
30110 OPEN #5, scr_512x12a0x204: PAPER #5,0: INK #5,6
30115 BORDER #5,1,6: CLS #5
30120 AT #5,0,16: PRINT #5," <f>orward  <b>ack  <p>rint "
30125 AT #5,0,44: PRINT #5," <n>ew file  <q>uit "
30130 CLS #4: dsp
30135 REPeat loop
30140 z=CODE(INKEY$(-1)): SELect ON z
30145  =70,102: CLS #4
30150           IF f+95>l
30155              snd: PRINT #4," Highest Display ": rst: GO TO 30135
30160           ELSE
30165              f=f+95: dsp
30170  = 66,98: CLS #4
30175           IF f=first
30180              snd: PRINT #4," Lowest Display ": rst: PAPER #4,4
30185           ELSE
30190              f=f-95: dsp
30205 REMark  ########## Other Commands  ##########
30210  =80,112: OPEN #3,ser: LIST #3, f TO f+95: CLOSE #3
30215  =78,110: WINDOW 512,256,0,0: PAPER 0: PAPER #2,0:INK #2,0: CLS
30220           DLINE first TO last: CLEAR: RUN
30225  =81,113: WINDOW 256,202,0,0: PAPER #2,6: INK #2,2
30230           BORDER 1,6,0: CLS: WINDOW 256,202,256,0: PAPER 2
30235           INK 6: BORDER 1,6,0: CLS: NEW
30240 END IF :  END SELect : END REPeat loop: NEW
30305 REMark  ##########  Procedures  ##########
30310 DEFine PROCedure snd: BEEP 2000,35: AT #4,18,30: PAPER #4,2
30315 END DEFine
30320 DEFine PROCedure rst: PAUSE 60: PAPER #4,4: CLS #4
30325 LIST #4, f TO f+95: END DEFine
30330 DEFine PROCedure dsp: LIST #4, f TO f+95: END DEFine
```

Durn! I forgot to renew my subscription to Update Magazine!

Comento:
Old folks never die.
They just Forget.
(Methuselah)

Please renew NOW. All 1988 subscriptions expire with this issue.
1990 year of issues begin with the next issue (Oct.88). Subscription $18.00 per year.

QL MEDIA MANAGERS

Bob Hartung
2416 N. County Line Rd., Huntertown, IN 46748

Several menuloader options are provided in this utility issue mdv/disk. The BOOT routine is a string-defined menu which contains only the file names for SuperBasic programs which will run from drive 1 by pressing just the one key 0-9 or a-z (lower-case only). If upper-case A is keyed the prompt will then ask for the code-key of the item to be added or changed. The selected string-definition line will then be brought into editing mode for insertion of the desired file name. Keying <sv> will re-save the revised menu as BOOT to drive 1. For TOOLKIT or TRUMP CARD users, if your file names contain only alphabetic characters (no numerals) line 140 may be revised to the default syntax, LRUN o$(sic). While QL file names are not limited in length, because of the display format this loader accepts names of up to 14 characters. The other loader routines accept up to 24.

When MENUload is installed as the BOOT routine it will save as cat_dat the directory of the media in drive 1, which will then be displayed as a menu on boot-up. Any lower-case key may be used to run the cursor through the page(s) listed. Keying <ENTER> will load the program at the cursor. The catalog data may be updated by keying upper-case N. Keying <sv> will save MENUload to drive 1 as BOOT. Since this loader displays all file names, for media containing a few master programs that draw extensively on data files it may be less confusing to use either the string-definition loader or else xMENUload. xMENUload is a version of MENUload which displays only those file names saved with the prefix x, ie: xNAME.

The QLdex routine is similar to the 2068 DOSDEX and allows saving directories from media in drive 1 to an index medium in drive 2 on which QLdex exists as a routine. Keying S will make consecutively numbered saves as cat_1, cat_2, etc. of the respective directories. N may be keyed to change the number of the Next directory to Load, or to update by a new Save. Keying either lower or upper case L will Load the Next-numbered catalog for display. Any other lower-case key moves the cursor through the page(s). Keying <ENTER> will LRUN the title at the cursor if the medium on which it exists is in drive 1. Like xMENUload, xQLdex displays only those file names saved with an x prefix.

TKTmenuMGR is a limited file-management utility for TOOLKIT II and TRUMP CARD users. Its file directory uses the same data as MENUload and xMENUload. TKTmanager is similar but uses the same directories as QLdex and xQLdex. Line 270 in these routines may be re-defined if desired to eliminate keying in the drive no. before each file loading.

If the printer is on-line, a hard copy will be made of directories and all HELP files as they are loaded in. The CAT_PRINT routine also prints out directories from either drive. Since CAT_PRINT saves then deletes a temporary cat_dta file, the media cannot be write-protected during this process. Printers that require an end-of-line CR may need either CHR$(10) or CHR$(13) inserted after, or instead of, each ending PRINT#n,... \ (back-slash character) in all program listings.

NOTE: Because the behavior of WHEN ERRor can be unpredictable at times, it was not used in these routines. If a program stops with an error it may be necessary to use both CLEAR and CLOSE #n for any channels remaining open. TOOLKIT II allows just CLOSE to close all channels from #3 upward. Still-open channels can result in "not found" and "changed or bad medium" reports.

QL_LLIST

This routine allows making an LLIST (printout) of any Basic listing, with universal selection of left margin, column-width, and number of lines per page for any printer. Listings longer than one page-length will be continued where the preceding LLIST column left off for printing side-by-side columns after the paper has been re-located in the printer. QL_LLIST also allows a printout of data from DATA_FILE (but not from ADDRfile1).

INFOWRITER

This is a quite primitive wordprocessor which was used to write the HELP info files for the various programs on the issue mdv/disk. Text written by INFOWRITER may be used for auto-addressed form letters with formal or informal salutations when used with MERGEWRITER and DATA_FILE.

Option 0 from the menu loads a temporary work-file, MYfile, which may be overwritten, or edited for one-time use. Selection of ALL-NEW brings up a prompt to ask if you want to erase any file already in RAM (files such as MYfile that are in drive media are unaffected).

LOAD or SAVE selections first display a directory of existing files so you may choose whether or not to overwrite existing files. (MYfile is always overwritten by a 0 selection at SAVE.)

A previously-saved file may be loaded in for LPRINT, editing, or adding to it at the next open line if ADD TO is selected. In edit mode, $clone deletes the displayed line and increments to the following line. $open will insert one open line preceding the displayed line. Use $to for backstepping and review of the format of previous lines.

The LPRINT routine contains a universal left-margin setting, but will require adaptation to your printer command codes if you desire to use fonts, etc. other than panel-selected or pre-set.

DATA_FILE

When first loaded, the asterisks appearing in the menu and the prompt in SET PRINTER serve as warnings that a file must first exist in DATA_FILE before it can be added to, sorted, printed, etc! When starting an all-new file the first prompt will ask if you want to erase the file in RAM. (Files in drive media are unaffected.) If y or Y is then entered you will be prompted to define (1) the number of fields to be formed for each file item, (2) the labels for these fields, and (3) the maximum number of characters allowed in each.

Since the number and size of defined fields determines how many array-items can be (automatically defined in RAM available, no larger values should be entered than necessary. Also, if you select the setprt option to LPRINT field 2 on the same line as field 1, ie, for title/first name then last name with one space between, you will want to label these fields accordingly. (This option also allows sorting and re-ordering by last names.)

In entry mode, $to as a first entry will back-step to re-enter ALL lines of that file item. $stop as a first entry will halt input and go to the main menu.

If LIST & EDIT is selected from the menu, each field of an item will be displayed with its label and a line no. UP/DOWN arrow keys scroll file items. Key ENTER to keep that item. $stop as an input will halt the EDIT mode and return to the main menu. The character % as a first entry will delete ALL fields of that item. The entry of a field-line code-character will allow revision of that field-line only, so for extensive revision of a file item it may be easier to enter % to delete it entirely, then return to the main menu with $stop, and select ADDITIONAL ENTRIES to re-enter it. Keying only ENTER in the edit REVISION mode will delete the line selected.

The SET PRINTER options allow (1) setting of a universal left margin which works with all printers, (2) the FIF2 option of printing fields 1 and 2 on the same line or not, and (3) the selection of how many fields will be printed, including 1 through the field no. selected. Phone nos. and/or search/sort codes may be stored in final field(s) and not printed. Your printer-code sequences to set font, etc. may be inserted in line 1150, eg:

```
1150 OPEN #7,ser1: PRINT #7,CHR$(27);"E": CLOSE #7
```

In search mode, if y or Y is selected at the LPRINT y/n? prompt, any found items will be sent to the printer in the format selected by SET PRINTER. The PAUSE condition in line 200 to allow for insertion of the next envelope for addressing may be deleted for continuous label rolls or envelopes in web carriers, and line 320 re-defined to insert the proper number of line feeds. The search word may be any consecutive set of characters. If n or N is selected at the LPRINT y/n? prompt, found items are displayed instead of going to the printer. Keying ENTER will continue the search, or y in response to the EDIT y? prompt will go to EDIT mode for that item, then resume the search mode.

When RE-ORDER is selected from the main menu, the options are (1) to select the field no. by which the sort will be done, and (2) set the number of beginning characters in that field which will be compared. Because the SUM of all character-codes is used by the QL for string comparisons, a re-ordering by more than, say, 4-5 character values may not produce the desired results in cases where one field with only 4 or 5 characters is compared with another that may have 15 or 20.

In the listing, note that line 5 defines as named variables the location of various routines used. This allows using these names instead of line nos. as in lines 520-610, and also allows ED name or EDIT name or AUTO name if you wish to examine a particular routine or experiment with making it a procedure. To avoid possible crashes, CLEAR before using TOOLKIT II ED mode with long listings containing large arrays, and BREAK after every screen-page of editing, especially on lines at the very bottom of the screen. Note that using RENUM may alter these locations so the program will not work unless line 5 is re-defined accordingly.

The displayed value for FREE MEMORY reflects the fact that after first dimensioning string arrays they all contain null characters, so allowance must be made for expansion of the array elements as they are written to.

ADDRFILE1

This is a specialized version of DATA_FILE and the MAIL FILE program that appeared in October 1988 UPDATE. It is useful for name, address, phone no. files that do not require more than 6 fields of 32 characters for each item (14 for state abbreviation + 2 spaces + up to 9-digit ZIP).

The fields are pre-defined so starting an all-new file takes you directly to the prompt for ITEM 1. Note that a first-entry of a number 1-5 followed by a first name will automatically do the printout with that title followed by the first name and the second name which has been entered in field-line 2. A 9 may be followed by any other title entered in full and/or a first name (no space between the number and first name).

Quick-entry of often-used P.O., state, and ZIPs may also be made with a single-digit entry to the P.O. field if lines 730-780 in the listing are revised to those you most frequently use. The entry for state/ZIP may then be skipped with just <ENTER>, or else P.O., state abbreviation, and ZIP entered in full if quick-entry digit is not used. Sorting and re-ordering may be done by either last names or by state/ZIP. See DATA_FILE editing info.

MERGEWRITER

This program is designed for merging text data formed by INFOWRITER with name and address files formed by DATA_FILE to print all or selected address headers with personalized formal or informal salutations picked up from the DATA_FILE data. Your letterhead is centered, and the current date is printed in full from data in DATE$, ie: October 1, 1989.

Field 1 from DATA_FILE may hold a title such as Mr. or Ms. followed by the first name, with the second name in field two for last-name re-ordering, or else title and full name may be in field one. Title, first name, last name will then be printed out on one line for the address heading. The title with last name or just the first name may be picked out of the file for either formal or informal salutation.

Page numbering with footer text may be used with multiple-page formfeeds. All address data may be used to automatically print letters and then labels in the same order, or else selected names may be searched out and printed with corresponding labels in the same order as the letters.

QSO_FILE

This is an adaptation of DATA_FILE for use as a logbook to enter date, time, frequency, signal strength, mode, comments and QSL response, with up to 30 QSOs each, into a name/QTH/callsign database. For an unexpanded QL, there is room for 81 stations per file (more with a reduced number of QSO entries, as in example included for 250 stations with 3 QSOs each). With RAM expansion, of course, the number of QSO entries for each may be increased.

Pre-set and labeled fields of 31-character length each are provided for first name, last name and callsign, address/QTH, and comments, with a 1-digit designation for QSL cards sent/received. By updating QSL information in EDIT mode, a sent/received percentage may be displayed. Also in EDIT mode or from SEARCH mode current QSO info may be entered for a selected name/callsign, or all data on file may be reviewed, updated, or edited by use of the four arrow-keys. Current date and time are entered automatically for a QSO entry.

Files may be re-ordered by any field. ~~All~~ search may be made for any consecutive set of characters in a name, callsign, or QTH. Found files may be either sent consecutively to the printer or displayed for review or adding a current QSO.

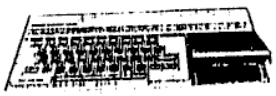
QSO_FILE was written at the suggestion of Alan Trainer, WB4NVC. For further help or info in adapting it or any of the other issue programs to your particular needs, please enclose SASE with your request to Bob Hartung. For updating previous versions with the latest revisions/corrections, please include return postage and a mdv cartridge containing your copy of the program(s).

QL CAPS LOCK CONTROL

While the multi-tasking CAPS LOCK indicator in April UPDATE is a very useful utility, there are times, as in menu selections, when all that is needed is a way of ensuring that CAPS LOCK is in a known state. Line 1 below sets it OFF, and line 3 sets it ON. Try RUN first to show this, then RUN 2.

-- Bob Hartung --

```
1 POKE_W 163976,0
2   PRINT #0;"CAPS LOCK ";: IF NOT PEEK_W(163976):
PRINT #0;"OFF": ELSE PRINT #0;"ON"
3 POKE_W 163976,-256
```



LKDOS TOOLS by the Half Dozen

```
 100 REM "LKDOS Print Tools #1-5"
 102 BORDER 0: PAPER 0: INK 7: CLS
 104 CLEAR : LET a=0: LET b=0: LET u$=""

9800 REM #1. Initialization Routine
9804 RANDOMIZE USR 100: OPEN #3,"lp": PR
INT #0;"Input Type of Interface"!"<1> Ae
rco-Oliger"!"<2> Tasman"!"<3> A & J": PA
USE 0: LET zz=CODE INKEY$-49: IF zz>2 TH
EN  CLS : GO TO 9804
9806 RANDOMIZE USR 100: POKE 16096,zz
9808 INPUT "Line Width?";Lw: RANDOMIZE U
SR 100: POKE 16090,Lw-1
9809 INPUT "Left Margin Tab?";Lm: RANDOM
IZE USR 100: POKE 16094,lm
9810 INPUT "<1> for Line Feed - <2> No L
f";Lf: IF Lf<1 OR Lf>2 THEN  GO TO 9810
9812 RANDOMIZE USR 100: POKE 16092,(10 A
ND Lf=1)+(0 AND Lf=2)
9814 PRINT #0;"Turn OFF PRINTER, Re-Set
Printerswitches, then TURN Back ON.    E
NTER when Done": PAUSE 0: LPRINT : CLS
9818 FOR n=1 TO 150 STEP 5: LPRINT "xxxx
 ";: NEXT n: LPRINT : REM Test print lin
es
9820 STOP
9822 REM #2. CAT PRINT *  Print a Disk C
atalog
9824 CLS : INPUT "Drive #";dn: RANDOMIZE
 USR 100: GO TO dn
9826 RANDOMIZE USR 100: POKE 16090,31: R
EM  * Set line to 32 Chrs
9850 RANDOMIZE USR 100: OPEN #2,"lp": RA
NDOMIZE USR 100: CAT "",: PRINT ': RANDO
MIZE USR 100: CLOSE #2
9860 PRINT AT 20,0;"Install new Diskette
, -OR-      ""Q"" to Quit, Any Key for N
ext": PAUSE 0: CLS : LET z$=INKEY$: IF z
$="Q" OR z$="q" THEN  GO TO 9864
9862 GO TO 9826
9864 STOP
9866
9868 REM ** Sending String Data to Disk
9869 REM #3. Create the U$ Data   first
* Use Caps Sym Shf to QUIT * Use ENTER f
or starting new line
9870 LET u$="": CLS : PRINT AT 20,0;"Tur
n Disk ON, Enter a file nbr  <1 to 5 onl
y)": INPUT zz: GO SUB 9970+zz: CLS : LET
 u$="": LET a=0: LET b=0: PRINT AT a,b;"
 ";:
9872 POKE 23611,220
9874 IF PEEK 23611<221 THEN  GO TO 9874
9876 LET c=PEEK 23560: POKE 23611,220
9878 IF c=12 THEN  GO TO 9890
9880 IF c=13 THEN  GO SUB 9940: GO SUB 9
902: GO TO 9872
9884 IF c=14 THEN  GO SUB 9902: GO TO 99
06
9885 PRINT AT 21,29;LEN u$;AT a,b;CHR$ c
;: LET b=b+1: IF b=32 THEN  GO SUB 9940
9886 PRINT AT a,b;"_";: LET u$=u$+CHR$ c

9888 GO TO 9872
9890 LET u$=u$( TO LEN u$-1): LET b=b-1:
 IF b=-1 THEN  LET b=31: LET a=a-1
9892 PRINT AT a,b;" ";: GO TO 9872
9902 PRINT #5;u$: LET u$="": PRINT AT 21
,29;"  ": RETURN
9906 CLS : PRINT AT 20,0;"<1> Create Mor
e Text"!"<2> QUIT": INPUT xx: IF xx=1 TH
EN  GO SUB 9940: GO TO 9872
```

A Fist Full of LKDOS TOOLS

As more TS-2068 users become LKDOS users the questions roll in. Some are working "Max DOS" (with Oliger Dos), but most are just getting their feet wetter with the rich LKDOS functions. Sometimes the process becomes a bit of a head scratching event. Update does not sell hair restoring lotion, but perhaps these utilities will have even better results.

The LLISTING for "LKDOS TOOLS" contains a group of utilities, each headed by REM statements. Each of these utilities may be plucked out to make individual sub-routines to MERGE to any BASIC program for their specific TOOL. There are no "ride-in Variables" and the utilities are independant. But there are a few vars that are re-used by the follow on tools.

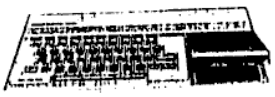
The first few lines are simple Screen set ups. Then Line 104 sets three vars that need "Fast Finding" by the utilities. When initialized FIRST in the vars file, each subsequent use makes the programs run faster than if a large file must be searched for the vars values.

TOOL #1, The LKDOS INITIALIZATION, begins at line 9800. When the computer is turned ON, LKDOS is asleep. The "Wake-Up" slap is <RANDOMIZE USR 100:> followed by a LK System Command. The LKDOS manual tells us that we can use <RANDOMIZE USR 100:OPEN#4,"dd"> and from then on we can use <PRINT#4:> followed by a System Command. However, as we get into "MAX DOS", the PRINT#4 causes some conflicts. Actually, I find that it is better to always just use RANDOMIZE USR 100 before the System Commands, and never <OPEN#4,"dd">. Later, we will see that having channel 4 OPEN for the PRINT#4 commands can result in "too many channels open" and cause some confusion.

So, as this programmers personal preference, these utilities wont use PRINT#4 at all! If you thought that PRINT#4 was an essential ingredient for the soup then this should convince you, as there are many many LKDOS system commands and all are preceeded by RANDOMIZE USR 100. AND, channel #4 remains closed throughout. You can corrolate the POKES in line 9804 through 9898 with the LKDOS "Version 3 EPROM" manual. <OPEN#3,lp> wakes up the PRINTER CODE in the LKDOS EPROM. At that time the key addresses in the LKDOS RAM chip are set to numbers that may or MAY NOT be suitable for your printer, or your desires. Beginning at line 9800 and ending at line 9814 is an initialization routine that will fit into your needs for all Basic Program.

The next TOOL begins at line 9818 and is simply a "Proof Test" of the initialization routine. But note the LPRINT at the end of line 9814. This is necessary after the initialization tools to reset the printer's carriage for the new LEFT MARGIN. Most printers requires an OFF and back ON for any switch settings that changes print style or margins. Line 9818 simply prints groups of five characters for you to easily check to see that your desired TAB and LINE WIDTH are correct. You ALWAYS POKE 16090 with ONE Number LESS than your desired line width. Also, I might mention that all of these system POKES require either RANDOMIZE USR 100 or PRINT #4 AHEAD of the POKE.

In fact, all LKDOS commands require the RANDOMIZE or PRINT #4. This is because the LKDOS system is operating in an EXTRA MEMORY BANK, "the DOS BANK". So, RANDOMIZE USR: or PRINT#4: causes a BANK SWITCHING caper in order to carry out the following LKDOS command. The LKDOS system opens up another bank of memory and uses 16K of that new bank of memory addresses. Later we may see more "chunks" of the LKDOS Bank used, but for now thats all of the noodles in that bowl of soup (16K). Half is the EPROM and the other half is RAM.



```
9908>RANDOMIZE USR 100:CLOSE #5:STOP
9918 STOP : REM #4. LPrint or View disk
file
9920 CLS : PRINT AT 20,0;"<1> View File
<2> LPRINT File": INPUT xx: IF xx=1 THEN
  GO TO 9928
9924 CLS : PRINT AT 20,0;"Set Printer ON
, Then ENTER": PAUSE 0: CLS : GO SUB 996
0: REM * Send data to printer instead of
 screen
9928 CLS : LET u$="": GO SUB 9980+zz: RE
M * Open disk file for INPUT to screen o
r printer
9930 ON ERR  GO TO 9946: INPUT #5;u$: RE
M * Bring in record from disk
9932 IF xx=1 THEN  PRINT u$:     REM * Pr
int this record and get next record from
 disk.
9934 IF xx=2 THEN  LPRINT u$: LET u$=""
9936 GO TO 9930
9940 LET a=a+1: LET b=0: IF a>21 THEN  P
OKE 23692,255: LET a=21
9942 PRINT : PRINT " ";: RETURN
9946 ON ERR  RESET : RANDOMIZE USR 100:
CLOSE #5
9950 STOP
9952 GO TO 9920
9956 REM #5. Gosub Routine for quick for
mat changes
9960 CLS : INPUT "Input TAB;";Lm: INPUT
"Line Width";Lw: RANDOMIZE USR 100: POKE
 16094,Lm: RANDOMIZE USR 100: POKE 16090
,Lw-1: LPRINT : RETURN
9970 REM * Tell me how to Concatenate a
number to OUT File?
9971 RANDOMIZE USR 100: OPEN #5,"Dta1.A$
  OUT ": RETURN
9972 RANDOMIZE USR 100: OPEN #5,"Dta2.A$
  OUT ": RETURN
9973 RANDOMIZE USR 100: OPEN #5,"Dta3.A$
  OUT ": RETURN
9974 RANDOMIZE USR 100: OPEN #5,"Dta4.A$
  OUT ": RETURN
9975 RANDOMIZE USR 100: OPEN #5,"Dta5.A$
  OUT ": RETURN
9981 RANDOMIZE USR 100: OPEN #5,"Dta1.A$
 IN ": RETURN
9982 RANDOMIZE USR 100: OPEN #5,"Dta2.A$
 IN ": RETURN
9983 RANDOMIZE USR 100: OPEN #5,"Dta3.A$
 IN ": RETURN
9984 RANDOMIZE USR 100: OPEN #5,"Dta4.A$
 IN ": RETURN
9985 RANDOMIZE USR 100: OPEN #5,"Dta5.A$
 IN ": RETURN
```

```
  5 REM "CODE FINDER"** Adaptable to all
 TS-2068 systems ** Finds start and end a
ddress, and byte length of any CODE progr
am.
  10 BORDER 0: PAPER 0: INK 7: CLS : INPU
T "Data Disk #?";dd: RANDOMIZE USR 100: G
O TO dd: RANDOMIZE USR 100: CAT "",: INPU
T "File name";z$: RANDOMIZE USR 100: LOAD
 z$CODE
  20 CLS : PRINT "Searching for Starting
Code Adr"'': FOR z=30000 TO 65535
  30 IF PEEK z<>0 THEN  PRINT "Code Start
 is ";z: LET aa=z: GO TO 50
  40 NEXT z
  50 PRINT ''"Searching for END Code Add
ress": FOR z=65367 TO z STEP -1
  60 IF PEEK z<>0 THEN  PRINT "End addres
s=";z: GO TO 80
  70 NEXT z
  80 PRINT ''"The byte length is ";z-aa:
```

The next LKDOS TOOL is at line 9822 and is a "CAT PRINTER". This lets you LPRINT a Disk Directory, another, and as many more as you would like to do, and from any drive in your system. You can get hard copies of all of your disks in the library with this CAT PRINT utility. Also, if you want to you can add <GOSUB 9960> just after the line number of line 9826. This would let you set up TAB for the Cat Print in order to print double column catalogs.

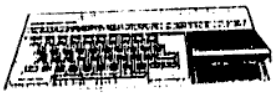
Tools #3 is a very versatile group of functions. First it uses some of the "TS-2068 DATA INPUT" program's techniques. Thats the neat program that was presented in the April 89 issue. Here it is abbreviated to provide a fairly fast input typing to a "Sequential File SAVER". Actually this set of functions is a "mini data base manager and printer", jerked together to demonstrate a method of creating managable SEQUENTIAL FILES in disk. The File Saver operates so slick that you dont even realize that the typed data is going into the disk file, and you wont believe that it is until proven by the next TOOL.

So, lets discuss whats happening between line 9869 and 9908. First, the DATA INPUT section operates to INPUT data from the keyboard. A "Character Counter" is printed at the bottom Right corner of the screen. You can create "LINES of data by watching the counter as you type. For example, if you plan to print lines of 64 characters, then as the counter reaches near 64, touch ENTER. The cursor springs down to the start of the next line--- B U T !! What you wont believe is that each time you touch ENTER, the previous line of data is socked away in a Sequential Disk File!! Actually ALL data ABOVE the current line being typed are "DEAD LINES". They have all been saved to disk, and are retained on screen just for your orientation.

Back at line 9890 a "File # is entered". Then a <GO SUB 9970 "OPENS a Sequential Disk File" with the name of "Dta" plus the file number. Example: file #1 will have the title of "Dta1.A$ OUT ". Thereafter each time ENTER is touched, the data in U$, the typing buffer, is saved to that file in disk. Lines 9872, 9874, and 9876 comprise the (now familiar) "Peek K State" INPUT typing function for reading the keyboard and assigning the key struck code to var c. If the key is DELETE, then line 9878 goes SUB to the DELETE Characer routine. If the key struck is ENTER, then line 9880 goes SUB to the Sequential File SAVER routine. After the line of data is SAVED the typing buffer U$ is zeroed and its back to the K-State for more characters.

Continuing- Then at line 9884, if CAPS-SYM SHIFT is the key, we have decided to QUIT, and the last line is SAVED at 9906. During input typing none of these escape routes are taken and line 9985 executes to do several things. First, the character counter prints the current length to keep you up-dated. Then the last character is printed to screen. Then the "print position variables a and b are up-dated. Then line 9986 prints the cursor and then Concatenates the last character to U$. At line 9888 the GO TO 9872 gets Peek K again for the next character. Now lets look back at lines 9880 and 9885. Both use <GO SUB 9940> which is to check to see if the screen is full.

If so, then at line 9940 we POKE 23692,255. This kills SCROLL and any subsequent line ENTERED will push up from the screen bottom, a SCROLL UP if you will. Line 9906 is the END ESCAPE routine. But it also gives you a choice of continuing input to the disk file. If you decide to quit then line 9908 does a V E R Y IMPORTANT function. It CLOSES the FILE with <RANDOMIZE USR 100:CLOSE#5>. This is absolutely essential when you finish inputing data to an open disk file. ANY LKDOS FILE is OPENED with <RANDOMIZE USR 100> and must be CLOSED in the same manner. Remember that LKDOS is operating from the "DOS BANK of memory" and



```
9990 REM *** LKDOS AUTO SAVE
    ROUTINE ***
9992 POKE 23730,255: POKE 23731,PEEK 2364
213
9994 RANDOMIZE USR 102: RUN
```

Two More TOOLS

The above two utilities are very useful to me. The LKDOS disk catalog does not contain the Start Address and END address or byte length of CODE FILES (Extension .C). So, one is left fumbling in the dark when one needs that information to peek a code file. If you can peek a code file you can convert the peeks to Characters (if the code represents character data). Or, if it is a CODE Program, the code can be disassembled. "CODE FINDER" does that for you.

Next, we have a nice little utility to AUTO SAVE a program. "Heck, thats easy to do with just RANDOMIZE USR 102, eh"? It sure is, but the SAVE takes about 7 cylinders of disk space! This is because an Auto Save SAVES every bit of RAM from PROG clear up to Ram Top. So an auto save actually saves 38K of stuff to disk, when the actual program may be only 2K in length and will fit in just ONE CYLINDER of the disk.

The "LKDOS AUTO SAVE" utility given above reduces RAM TOP to about 750 bytes above the basic program and its variable file. Then it saves the program so that it auto runs by holding the ENTER key as the computer is turned on.

There is some rationale about Auto Running programs with LKDOS. We will attack it this this way. We reduce RAM TOP to make the program shorter, thus taking up less disk space. BUT- when the program is reloaded, RAM TOP is STILL REDUCED. The result is that the program will Auto Boot O.K., but then may have a severe shortage FREE MEMORY for operation. If we reduce FREE (by setting Ram Top DOWN before the save) then it could be that the program wont run after auto loading. The way to fix this is to "prepare the program before saving it". We do that by installing a bit of programming in the FIRST convenient program line of the program to be saved.

There are two addresses, the contents of which controls Ram Top. They are addresses 23730 and 23731. Upon start up these two addresses have these values: 23730=87 and 23731=255. So, we should POKE these value to the two addresses EARLY in the auto run program. We do that by adding to the first operating line of the program: <POKE 23731,255:POKE 23730,87>. Then when the program auto loads Ram Top will be set to 65367, its normal setting.

---

wont operate without RANDOMIZE USR 100.

TOOL #4, is the "View-PRINT" for Sequential Disk Files. These functions begin at line 9918 and end at line 9946. First, at line 9918, the STOP is to keep this package of functions from operating after the previous routine is finished. So, if you are operating both together, the STOP will be reported. Then to enter the routines type <CONT ENTER>. Line 9920 gives the choices of Viewing the Disk File or Printing it. We will skip the rationale about screen viewing and work with PRINTING. Line 9924 goes SUB to a neat little "PRINTER FORMAT" routine at line 9960, where you can format the TAB and Line width.

Then line 9928 does a little shenanagin of selectively OPENING a previously closed file in disk. Variable ZZ is carried over from the INPUT TOOL section (line 9870). Its value is between 1 and 5. So the proper line for the GO SUB is already fixed. This group of lines are the "OPEN#5 "IN" group". The objective is to OPEN the disk file and read the file back to the printer. A little parenthetical comment is appropriate. I couldn't find a key to Concatenate the titles with a number for either the OUT titles or the IN titles. This leaves the door open for someone to feed back the solution.

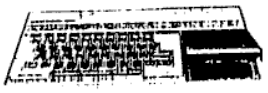
At line 9930 the <INPUT #5;u$> "reads the disk file that has been opened. The disk file contains as many "RECORDS" as were previously separated by "ENTER". Saying that in a different way; Each time you SEND data to a OPEN FILE, it forms a new RECORD. Remember that earlier we sent the "lines of data" to disk each time we touched ENTER. So, each line became a separate RECORD, and now we are READING the records back from disk, one at a time. MOVING on to line 9932 and 9934, variable xx=2 for PRINTING, so line 9934 executes to LPRINT U$, which has the current RECORD from disk.

Then Line 9936 goes back to line 9930 to get the next record. There is one problem to overcome. We are in a LOOP here, returning to line 9930 to keep on reading in records from the disk file. When there are no more records, the system will cause a report code "no more records" and stop. So, we place <ON ERR GO TO 9946> at the beginning of line 9930. Then when there is an attempt to read beyond the last record, the ON ERR will trip and line 9946 will execute.

When line 9946 executes, every record of the disk file has been prinied. The ON ERR is RESET, and the disk file is CLOSED. And that is the end of this haranguing group of epistles (except for a few closing comments).

Sequential Files seemed to be a very complicated part of LKDOS to me. So, I pushed it aside until I could get rountuit. Actually, once one gets used to "OPENING" and CLOSING FILES, which is the principal bug-aboo, sequential files offer great flexibility of managing data. The file handling is much faster than I thought possible. The principal advantage is the TINY amount of computer memory that is used. One can just keep on building a disk file to an extremely large size, much larger than can be handled within FREE RAM memory. These utilities points out just one method of data management with sequential files using short "line length" records. A record can be from a single character in length to several thousand characters.

Each individual save to disk is a separate RECORD. So, to build large records, one just keeps on typing and when finished, do the SAVE with <PRINT #5;"data" or U$>. Dont forget to CLOSE the file when finished, either sending or retrieving data. Have fun! -BJ_



UPDATE ANNOUNCES "Daisy.B6" For TS-2068

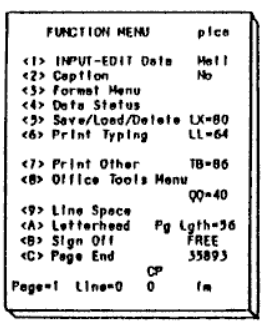
Daisy.B6 is a hi-bred update of the Smart Text software ensemble. Most of the Daisy program remains in disk with MERGE being used extensively to bring in parts of the program when called for by menu electives. This concept, along with 29 menus and information screens in disk, allow the program length to exceed 75K and many more functions to be provided. This is the largest software ever produced for the TS-2068 and it encompasses a full range of Administrative functions. Daisy has five different Data Bases. Text files, mail files, outline files, D.base-1 files, and sequential files, are all coordinated by either manual Menu electives for LOAD SAVE, or by selection of automatic sequences that saves data by discrete file name or loads data files for automatic printing. Any number of disk data files may be printed "in sequence" as a continuous document or letter. Mail files may be used for printing of labels or envelopes. Literally dozens of other printing products are available via menu electives.

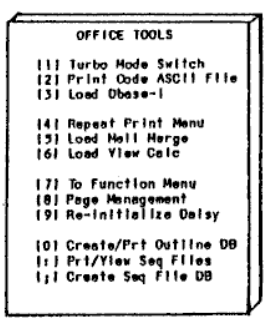
Daisy's menus allow full versatility of printing styles, formats and line lengths. The printer control group is for either Dot Matrix Epson Compatable printers or for any kind of Daisy Wheel printer. This 'printer control group' can easily be re-configured for any Centronics printer. As programmed, Daisy prints only with the Aerco or Oliger printer interface and will require some line number revision to print with A&J or Tasman interfaces.

In addition to printing its own data base products the Daisy menu selections provides for quick print out of ANY kind of data file created by other softwares and recorded as an ASCII Code files. This includes Mscript, Tasword, and Modem download down-load files. Two other softwares are integrated by Daisy. Menu electives select either D.base-1 or the View Calc program for operation. The data products of D.base-1 are used by Daisy for print out. Each of those programs returns to Daisy upon exit of the program.

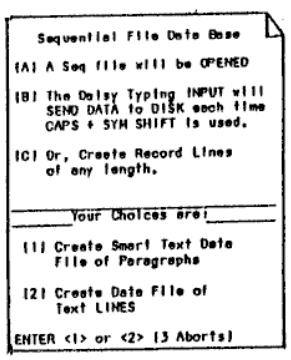
Daisy creates its own text files via fast input typing and FULL EDIT. Two types of Text files are optional. 1. A flexible Character Array file consisting of individually managed data groups. 2. Sequential files using direct coordination with disk. Both types of files may be reloaded from disk for full editing and modification, and both types of files may be sent to the printer for print-out in the format selected at menu.

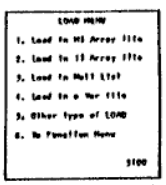
Following are just SOME of the Daisy Menus and screens, given to provide an overview of the software.

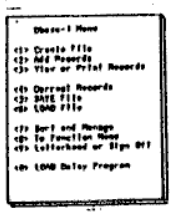




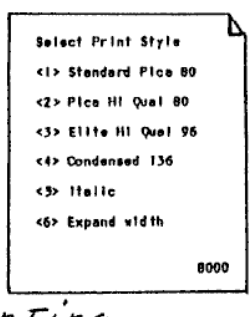
DISK DATA BASE MANAGEMENT


Sequential File Data Base
[A] A Seq file will be OPENED
[B] The Daisy Typing INPUT will SEND DATA to DISK each time CAPS + SYM SHIFT is used.
[C] Or, Create Record Lines of any length.
Your Choices are:
[1] Create Smart Text Data File of Paragraphs
[2] Create Data File of Text LINES
ENTER <1> or <2> [3 Aborts]

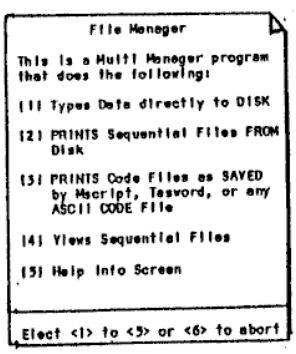

LOAD MENU
1. Load in MI Array file
2. Load in II Array file
3. Load in Mail List
4. Load in a Var file
5. Other type of LOAD
6. To Function Menu
5100

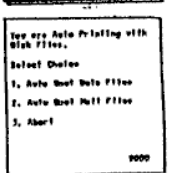


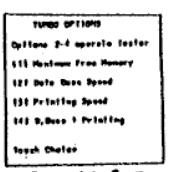
Actually There ARE quite a few MORE !



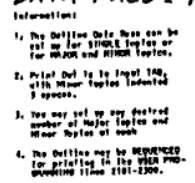
AUTOMATIC PRINTING


File Manager
This is a Multi Manager program that does the following:
[1] Types Data directly to DISK
[2] PRINTS Sequential Files FROM Disk
[3] PRINTS Code Files as SAVED by Mscript, Tasword, or any ASCII CODE File
[4] Views Sequential Files
[5] Help Info Screen
Elect <1> to <5> or <6> to abort

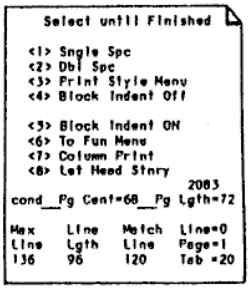


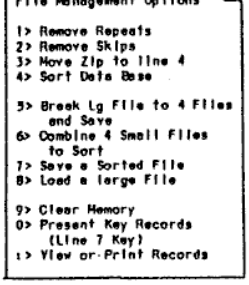


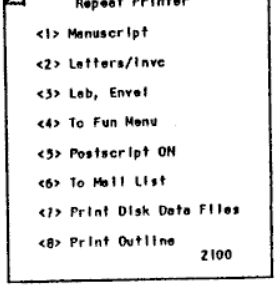
PRINTS OTHER SOFTWARE DATA Files - ASCII FILES

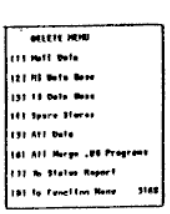

Information!
1. The Outline Data Base can be set up for SINGLE topics or for MAJOR and MINOR topics.
2. Print Out is to input 140, with Minor topics indented 3 spaces.
3. You may set up any desired number of Major topics and Minor topics at each
4. The Outline may be RENUMBERED for printing in the USER PRO-GRAMMING time 2101-2300.
5. The data base is "MAIL error" for MAJORS and "MAIL error" for MINORS. (ENTER)

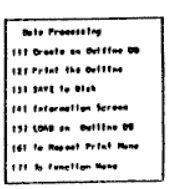
Sequential File Printer
[1] A Seq file will be OPENED
[2] The Daisy Word Proc will Print the data as per the FORMAT established.
[3] When finished the file will be CLOSED.
ENTER a file Number to Access the data- - -
- OR just ENTER to Abort

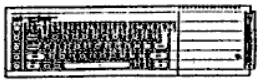



File Management Options
1> Remove Repeats
2> Remove Skips
3> Move Zip to line 4
4> Sort Data Base
5> Break Lg File to 4 Files and Save
6> Combine 4 Small Files to Sort
7> Save a Sorted File
8> Load a large File
9> Clear Memory
0> Present Key Records (Line 7 Key)
:> View or Print Records






Data Processing
[1] Create an Outline DB
[2] Print the Outline
[3] SAVE to Disk
[4] Information Screen
[5] LOAD an Outline DB
[6] To Repeat Print Menu
[7] To Function Menu



NEW Issue Disks for the Q.L.

NEW QL ISSUE DISKS FOR JULY 89

THREE (3) new QL DISKS or MDVs are offered! I am typing this column with one program that is in the "CABLE C" Disk. This program is an adaptation of one given by Simon Goodwin in QL World, by Phil Spink, and improved by Bill Cable. The best way to describe it is to tell you how it is operating. I am working with Quill to type and print this. There are four programs Multitasking at the same time. This is all four of the PSION suite of programs. To change to Archive, I just touch <CTRL> C and then press <SHIFT F5>. The first action shifts from QUILL into ARCHIVE. The second action Updates the screen to the ARCHIVE screen. Another shift gets ABACUS, another gets the third PSION program, etc, until QUILL comes up again. What does this do for us? The existing data file of each program remains in memory, and you dont have to SAVE or RE-LOAD anything to go from one software to another. They are all operating at the same time (multitasking). Most other EXECutable Super Basic programs can be added to the multitasking ensemble, for up to 10 programs running at the same time with Trump, or fewer in an un-expanded QL. The number depends upon the individual programs' length and how much memory each needs for data alloction. Documentation is furnished with the Issue Disk or MDV.

The second program in the CABLE C disk is "Filer", the key-in project given in this issue. 'Filer', a program to manage Archive Files is fully described in the text of the article by Bill Cable, and has more instructions in Doc Files.

Order the "CABLE C" disk and get both programs. $22.00 postpaid. If ordering for Micro Drive, send a Formatted MDV cart for dubbing. This is easily a $50.00 value.

The "CABLE B" Issue Disk MDV:

The main feature of this series contains a group of Procedures that will allow you to design, create, and manage, your own ARCHIVE Data Bases in an extremely easy environment. The program is aptly named "Dbeasy". You only need to load ARCHIVE and type 'RUN dbeasy ENTER". The rest is just following Menus that allow you to create and manage a data base "shaped to your needs". Anything that you would keep on file cards or in listings can be easily handled. For example, you can establish many records, each having 10 lines of 64 characters each (more or fewer). Once you input the data to the records you select, search, and order the records in a wide variety of ways via Menu electives. You can Print Out the records in a wide variety of ways. An extensive Doc file of instructions teach you how.

This versatile program is supplied post paid on either floppy disk or MDV for $22.00 (at least a $50.00 value). If ordering for MDV please send a Mdv Cart for dubbing. (Ask for the CABLE B Disk).

THE HARTUNG 6-3 DISK

Bob Hartung gives us several very useful programs, plus a menagerie of useful Disk Management utilities. The utilities are given in this and past issues of Update as key in projects. The disk directory follows. ** indicates PROGRAMS. * indictes utilities. In addition, Update has added "CAPS", the neat Caps Indicator given in a previous issue of Update. To make CAPS multitask with any operating program, type: exec mdv1_caps ENTER. Or the same may be entered into any program line to have CAPS multitask with the program. (A Caps Lock Indicator in the RH Top corner of the screen). The HARTUNG V6 Issue Disk Directory:

```
HARTUNG 6_3
18/216 Sectors
BOOT
Data_file **
Infowriter **
infohelp
helpl
help
QLDEX *
CAT_PRINT *
Addrfilel **
QLLLIST *
MFinfo
ADinfo
QL_VAL *
MENUload *
xQL_DEX *
xMENUload *
TKImanager *
TKT Menu Mgr *
Mergewriter **
mrginfo
dfile test
CAT_DAT *
CAT_1
testfilel
myfile
QSO_FILE **
QSO_FILE2 **
QSOinfo
CAPS *
```

How can one tag a value to such a large ensemble? $22.00 postpaid. Ask for the Hartung 6-3 Issue Disk. If ordering for Micro Drive, send a formatted MDV cart.

NEW TS-2068 ISSUE DISKS !

daisy

"Daisy.B6": This new software started off as "Smart Text" and has been used for the past couple of years to publish Update Magazine. "Leave good enough alone" is an adage disregarded. Now Daisy.B6 takes on a new look with the extensive use of the MERGE function to boot in many new functions that have been added. This multiple Data Base Manager is used to create a wide variety of useful data bases and then print them in the user's choice of formats and print styles.

"Automation" is the theme of Daisy. 29 menus and set-up instructions boot in from disk as called for by the operation in progress. Daisy is designed to do everything that one needs a computer for, except game playing. The print products are so flexible that Daisy prints straight formatted pages, automatic Columns, or offsets. Letters are automatically formatted. Repeat letters to mailing lists are automatic, and Daisy creates its own mailing list data base.. Label printing is a breeze. Included is likely the best Printer Control Group ever produced, designed for controlling Epson Compatable printers and all Daisy Wheel printers. Easily adapts for any other type of printer except Serial printers

The Daisy software MANAGES disk drive as if the disk were a part of the TS-2068 Memory. The program length is approximately 75K, with most of it in disk. Daisy is the largest software likely to ever be produced for the TS-2068. It has more functions than any TWO of the QL PSION softwares combined, or the IBM Word Perfect that sells for over a C note. It will take several months of steady use to learn all of the Daisy functions, but one can start right off doing easy letters. Daisy for LKDOS is now available. The Oliger DOS version will be finished by August 1, 89, and the FD-68 version by Sept. 89.

** - * - * - * - * - * - * - * - **

V·Calc··dd

* View Calc For Disk Systems *

The "Mitchell VC" Issue Disk

We have always needed a View Calc program that works with Disk Drive. The TS-2068 View Calc program has been around for several years, but most of us quit using it because Cassette operation is so slow. Now Bob Mitchell has converted View Calc for LKDOS operation. Update will do a Oliger and FD-68 version.

Bob adds Help Files, and a Compiled Machine code "Spreadsheet Printer", plus disk save and load. The latter makes it easy to manage view calc files without having to wait for the moon to come up while a file loads from cassette. This is a snappy program that one can now use to advantage. Use the article in this issue for documentation, plus your old View Calc manual.

Available now for LKDOS. Oliger DOS version by August, and FD-68 version by September.

** - ** - ** - ** - ** - ** - ** - ** - **

d..base..1

* The New "D.Base-1" Software for Sinclair Computers *

An Issue Disk Program

"D.base-1" is described fully in the key-in article given in this issue. The choice is given to key in the program or order it as an issue Disk, and get some added utilities. This multiple data base software creates and manages files of records. Use it for almost any conceivable need, from mailing lists, inventory lists, daily diary, family records, invoices, making race track tout sheets, outlines, appointment schedules, genealogy, stamp collection records, or whatever. An unprecedented variety of sorting is provided. Sort by first group of any line or last group of any line. The data base is so very flexible that any size or shape of records may be designed.

A wide variety of Print-outs allow one to print labels, full records, or partial records. Page control with page numbering is optional. Disk management allows the files to be recalled for additions, editing, re-sorting, clean-up of repeats, and elimination of skips in the records. A built in letterhead and signature (sign off) allows invoices to be managed, or the production of letterhead stationary. A separate data base is "Outlines" that allows one to create flexible outlines as used in school work or descriptive writing. An outline printer is also included. A very flexible data base program that includes managed printing and disk drive coordination.

D.Base-1 is available now for LKDOS and Oliger DOS. Target date for FD-68 version is August 89.

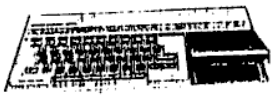
** - ** - ** - ** - ** - ** - ** - ** - **

* EXCLUSIVELY FOR LKDOS *

1. The UPDATE COLLECTION: This Issue Disk contains all of the LKDOS Utilities given in the magazine during the 1989 year of publication, including this issue. The major sortwares are not included, just utilities. This is quite a voluminous group of utilities for LKDOS users.. Both Oliger and FD-68 Collections are in works. No favoritism here, just have my plate full right now.

2. NOT AN UPDATE ISSUE DISK: But every LKDOS user should have it! "The OMNIBUS DISK" for LKDOS, is a collection by Bob Mitchell, 20 Wild Briarway, Willodale, Ontario M2J 2L2, CANADA. Update does not know the price, but has examined the Omnibus Disk. It is chuck full of Excellent utilities for LKDOS. There are only a few programs that are duplicates of the "Update Collection". Write for price and availability.

Ordering: Please specify your type of Disk System, and whether for DS-DD 40 track or DS-DD 80 track Drive. Issue disks are $22.00 each Post Paid, and are guaranteed to please or full refund.



TS-2068 Data Input (re-visited)

OR- "A new Hiding Place for Code"

The program given in the April 89 issue titled "TS-2068 INPUT" has a machine code program "Concatm.C1" (By Don Lamen) which provides fast input typing and concatenation of the input data to a typing buffer. That program works just fine, but we used the TS-2040 Print Buffer area for storing the code program. I ask Don to provide the code for a new storage place in memory starting at Address 24311. Following is the Z80 assembly instruction program for the new "concat.C1" code program. Then a BASIC program is given to POKE the Code to memory and then SAVE it to your disk. This LOADER/SAVER program can be used for universal purpose to install any code program to memory and check-sum the program. When you run it, INPUT 24311 for starting address and 202 for Byte length. Then for the SAVE, input "concat.C1" as the title to save to disk.

Next, to operate the "TS-2068 DATA INPUT" program as given in the April 89 issue, with this newly located code, change Line number 150, 167, 265, and 390 to "RANDOMIZE USR 24355". Also change line number 158 PEEKS to <LET a=PEEK 24314> <LET b=PEEK 24313> <LET a1=PEEK 24311+256*PEEK 24312>. Then the "TS-2068 DATA INPUT" program will run with this new Code Location. (After it is saved by the "LOADER PROGRAM LISTING" given herein).

This new location of the Concat.C1" code program opens up the Printer Buffer area for its normal use, and for use of the Oliger CPI printer code. Also the new location does not cost any FREE memory, and is safe from being overwritten by an operating program.

The LOADER PROGRAM LISTING assumes that the user has initialized the DOS before using RUN to operate the loader. Also line number 250 is set up for a LKDOS SAVE of the concat.C1 Code file. If using another DOS replace the RANDOMIZE USR 100 and SAVE with the proper syntax for the DOS. Once the LOADER program is keyed in it is ready to operate with <RUN>. It first pokes the code to the memory addresses, then checks the sum. If the sum is as given in line 2022 (and input at the prompt), the LOADER program proceeds to the Disk SAVE line at line 250 and SAVES the code program to disk. You should input "concat.C1" as the title for SAVE and input 24311 as the starting address and 202 as the byte length.

Before the code SAVES to Disk, Line number 230 peeks the code back from memory and checks the SUM to make sure that the DATA lines have been keyed in correctly. If not you get the long toot and will have to re-check your work. I would suggest that you use a different diskette than the one that has the original concat.C1 program. Then transfer the original "DATA INPUT" Basic program to this new disk. This way you will have two DATA INPUT programs, one working with the old code and one working with the new. We are spending this extra space in Update because the DATA INPUT program given in the April issue and amended here is the best "Front End" for a Data Base program that has ever been given for the TS-2068. Everyone should have it.

Dear Bill,

Here is the update on the "concat" code.

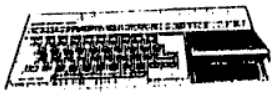
The new entry points are:

Variable 1, 24311 (2 bytes) --- LEN u$ (BASIC VAR A1)
Variable 2, 24314 (1 byte) --- LIN (BASIC VAR A)
Variable 3, 24313 (1 byte) --- COL (BASIC VAR B)
Variable 4, 24315 (1 byte) --- CHAR. (BASIC VAR C)
Entry point 1, Print the last 148 characters.
Entry point 2, Concatenate

I have added six new bytes from 5F03h to 5F08h to prevent a crash in the event of a missing u$, if there is no u$ a " 2/n Variable not found " error report will be printed.

CODE DISASSEMBLY

```
24311  5EF7 0000        VAR-1:   LEN u$       ; (A1)
24313  5EF9 0000        VAR-2:   SPOSN        ; COL (B) / LIN (A)
24315  5EFB 00          VAR-3:   CHAR.        ; (C)
                                                      5 bytes

24316  5EFC 2A4B5C    FIND-u$:   LD HL, (VARS)
            7E            CHECK:  LD A, (HL)
            FE55                  CP 55         ; Is it u$
            C8                    RET Z         ; If yes return
       5F03 FE80                  CP 80         ; End of VARS ?
            2002                  JR NZ, NEXT-ONE ; NO it's OK
            CF01                  RST 08  DEFB:'2' ; Not found
            CD2017    NEXT-ONE:   CALL 1720, NEXT-ONE
            EB                    EX DE, HL
            18F0                  JR CHECK
                                                     19 bytes

24335  5F0F CDA608        CLS:    CALL 08A6, CLS
            010000                LD BC, 0000
                                                      6 bytes

24341  5F15 213B5C     PRINT-AT:  LD HL, 5C3B, FLAGS
            CB8E                  RES 1, (HL)
            213C5C                LD HL, 5C3C, TY-FLAG
            CB86                  RES 0, (HL)
            CD8205                CALL 05B2, PRINT-AT
            C9                    RET
                                                     14 bytes

Entry point 1: (PRINT LAST 148 CHARS.)
24355  5F23 CD0F5F  PRINT-148:   CALL 5F0F, CLS
            CDFC5E                CALL 5EFC, FIND-u$
            23                    INC HL
            4E                    LD C, (HL)
            23                    INC HL
            46                    LD B, (HL)
            ED43F75E              LD (VAR-1), BC   ; LEN u$
            23                    INC HL
            EB                    EX DE, HL
            219400                LD HL, 0094      ; 148 dec.
            A7                    AND A            ; Reset cy
            ED42                  SBC HL, BC
            09                    ADD HL, BC
            3008                  JR NC, PRINT$    ; If LEN u$
            EB                    EX DE, HL          <= 148 then
            09                    ADD HL, BC         print u$.
            A7                    AND A            ; If LEN U$
            ED52                  SBC HL, DE         > 148 then
            EB                    EX DE, HL          print last
            44                    LD B, H            148 chars.
            4D                    LD C, L
24388  5F44 CD0B21      PRINT$:   CALL 210B, PR-STRING
            ED4B885C              LD BC, (SPOSN)
            3E18                  LD A, 18         ; 24 dec.
            90                    SUB B
            47                    LD B, A
            3E21                  LD A, 21         ; 33 dec.
            91                    SUB C
            4F                    LD C, A
            FE20                  CP 20            ; New line ?
            2003                  JR NZ, STORE
            0E00                  LD C, 00
            04                    INC B
24410  5F5A ED43F95E    STORE:    LD (VAR-2), BC   ; Store param.
            CD155F                CALL 5F15, PR-AT
            3E5F                  LD A, 5F         ; Cursor
            D7                    RST 10           ; Print cursor
                                                     65 bytes

Entry point 2: (CONCATENATE)
24420  5F64 3A3B5C      CONCAT:   LD A, (FLAGS)
            FEDD                  CP DD            ; 221 dec.
            38F9                  JR C, CONCAT     ; Wait for key
            3A085C                LD A, (LAST-K)
            5F                    LD E, A     ; Save char. in E.
```



```
                3EDC            LD A, DC        ; 220 dec.
                323B5C          LD (FLAGS), A
                7B              LD A, E
                FE20            CP20            ; Is it < 32 ?
                3004            JR NC, CONT     ; No it's OK.
                219E00          LD HL,009E  ; If < 32 dec. then
                                            go to line 158 in
                                            BASIC.
24444  5F70 CDD616             CALL 16D6, LINE-ADDR.
                22595C          LD (NXTLIN), HL
                C9              RET
24451  5F83 32FB5E     CONT:    LD (VAR-3), A   ; Save char.
                CDFC5E          CALL 5EFC, FIND-u$
                23              INC HL
                4E              LD C, (HL)
                34              INC (HL)
                23              INC HL
                46              LD B, (HL)
                2001            JR NZ, ADVAN
                34              INC (HL)
                09       ADVAN: ADD HL, BC      ; HL-> to last
                CDB81Z          CALL 12B8, ONE-SPACE     char.
24469  5F95 3AFB5E              LD A, (VAR-3)   ; Get char.
                23              INC HL
                23              INC HL
                77              LD (HL), A      ; Add it to u$
24475  5F9B ED4BF95E     PR-2:  LD BC, (VAR-2)
                CD155F          CALL 5F15, PR-AT
                3AFB5E          LD A, (VAR-3)   ; Get char.
                D7              RST 10          ; Print it
                3E5F            LD A, 5F        ; Cursor
                D7              RST 10          ; Print cursor
24489  5FA9 2AF95E              LD HL, (VAR-2)
                2C              INC L
                7D              LD A, L
                FE20            CP 20           ; Is it 32 dec
                2003            JR NZ, UPDATE
                2E00            LD L, 00
                24              INC H
24501  5FB5 22F95E     UPDATE:  LD (VAR-2), HL  ; 'AT' Param.
                2AF75E          LD HL, (VAR-1)  ; LEN u$e
                23              INC HL
                22F75E          LD (VAR-1), HL
                18A3            JR CONCAT
                                                93 bytes.

24513  5FC1 NEXT BYTE                  TATAL 202 bytes
```

I expect to have the machine code routines for Database-I ready in time for the October issue of UPDATE.

For Better Computing,

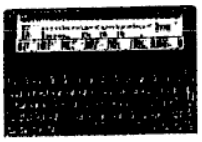
Don Lamen
SINCUS

**** SINC LINK ****

During this uncertain time about Sinclair Publications the Ontario TSUG is rising to the occasion by increasing the size and content of SINC LINK, their magazine size newsletter. The MAY-JUNE 89 issue is 24 pages and is chuck full of detail. Subscription is $12.00 (U.S). Enclose US Cash bills or Intl. Money order. (US checks costs for processing).
Sinc Link, 14 Richome Court, Scarborough, Ontario M1K 2Y1.

```
200>REM *** LOADER PROGRAM
           LISTING **
 210 CLS : INPUT "Input Check Su
m of Program";ck: INPUT "Input S
tart Addres to SAVE Code ";x1: I
NPUT "Input Byte Length:";x2
 220 RESTORE 2000: LET x=0: FOR
n=x1 TO x1+x2-1: READ a: LET x=x
+a:: POKE n,a: REM *Poker*
 225 NEXT n
 230 LET b=0: FOR n=x1 TO x1+x2-
1: LET b=b+PEEK n: REM * Get Act
ual Sum *
 232 NEXT n
 234 IF b=ck THEN  GO TO 250
 240 BEEP 2,6: PRINT "ERROR!  Re
check work.": STOP
 250 CLS : INPUT "Key In a Code
Name To SAVE.      Include Extens
ion-";a$: INPUT "Input Starting
Address for SAVE ";x: RANDOMIZE
USR 100: SAVE a$CODE x,x2
 260 CLS : PRINT AT 10,0;"The co
de program has been SAVED at add
ress ";x: STOP
2000 DATA 0,0,0,0,0,0,42,75,92,126
,254,85,200,254,128,32,2,207,1,2
05,32
2002 DATA 23,235,24,240,205,166,
8,1,0,0,33,59,92,203,142,33,60,9
2,203,134
2004 DATA 205,178,5,201,205,15,9
5,205,252,94,35,78,35,70,237,67,
247,94,35,235
2006 DATA 33,148,0,167,237,66,9,
48,8,235,9,167,237,82,235,68,77,
205,219,33
2008 DATA 237,75,136,92,62,24,14
4,71,62,33,145,79,254,32,32,3,14
,0,4,237
2010 DATA 67,249,94,205,21,95,62
,95,215,58,59,92,254,221,56,249,
58,8,92,95
2012 DATA 62,220,50,59,92,123,25
4,32,48,10,33,158,0,205,214,22,3
4,85,92,201
2014 DATA 50,251,94,205,252,94,3
5,78,52,35,70,32,1,52,9,205,184,
18,58,251
2016 DATA 94,35,35,119,237,75,24
9,94,205,21,95,58,251,94,215,62,
95,215,42,249
2018 DATA 94,44,125,254,32,32,3,
46,0,36,34,249,94,42,247,94,35,3
4,247,94
2020 DATA 24,163,0
2022 REM Check sum=21223
```

This Loader Program Listing was created in its entirety, including the DATA lines, by the "Mlz.B4" program, given on page 4.



DOMINO CUBES 212 971 5638
355 WEST 39th ST.
NEW YORK CITY 10018-1401
PRESIDENT.....MIKE FINK

THE Z-88 STARTS AT $488; THEN DROPS $6 FOR EACH $100 WORTH OF RELATED ACCESSORIES YOU ORDER-STOPS AT $428

PRICE LIST

1.	THE Z-88 PORTABLE COMPUTER	$488
2.	32K RAM OR EPROM CARDS.	$ 45
3.	128K RAM OR EPROM CARDS.	$ 95
4.	512K RAM CARDS	$395
5.	PARALLEL PRINTER CABLE	$ 62
6.	SERIAL PRINTER CABLE (*)	$ 22
7.	PC LINK - IBM	$ 75
8.	PC LINK - MACINTOSH	$125
9.	PC LINK - QL.	$ 45
10.	EPROM ERASER - 2 CARDS	$ 75
11.	EPROM ERASER - 4 CARDS	$ 95
12.	A/C ADAPTER: Z-88, WALL.	$ 15
13.	A/C ADAPTER: Z-88, CAR	$ 15
14.	MY MANUAL "A" FREE WITH Z-88	$ 10
15.	MY MANUAL "B"	$ 20
16.	SQUARE PIN PRINTER 135 CPS	$300
17.	SQUARE PIN PRINTER 700 CPS	$700
18.	PORTABLE! INK-JET PRINTER 180 CPS	$405
19.	CARRYING CASE "A"	$ 25
20.	CARRYING CASE "B"	$ 85
21.	202K PORTABLE DISK DRIVE +	$420
22.	20 HR. AA BATTERIES (4)	$4.5
23.	40 HR. AA BATTERIES (4)	$ 14
24.	5 NI-CAD 'C' BATTERIES	$ 25
25.	1200 BAUD MODEM PKG	$275
26.	3.5" DISKS, 10 PACK	$ 30
27.	RIBBON-FILM C130BK PRINTER	$5.7
28.	INK-JET INK, PLAIN or INK-JET	$ 13
29.	INK-JET STAND	$ 15
30.	COLORED INK (R-G-B)	$ 15

FREE DELIVERY UNLESS YOU ORDER A PRINTER, THEN ADD $15

♣ ♠ ♥ ♦ ♣ ♠ ♥

THE ONLY DEALER IN THE WORLD WHO GIVES YOU AN *UNDERSTANDABLE* BEGINNERS INSTRUCTION MANUAL!! AND THE ONLY DEALER SELLING AN AFFORDABLE EPROM ERASER, AND THE PLACE TO BUY THE PORTABLE DISK DRIVE TO SAVE LOADS OF MON$Y, FREE LOTTO PROGRAM WITH Z-88 PURCHASE, ALSO THE ONLY PERSON ON THE PLANET WHO CAN SHOW YOU HOW TO COLLATE, AND PRINT MULTIPLE COPIES OF MULTIPLE FILES EASILY!!! *WHILE YOU ARE ASLEEP OR AT THE MOVIES!!*

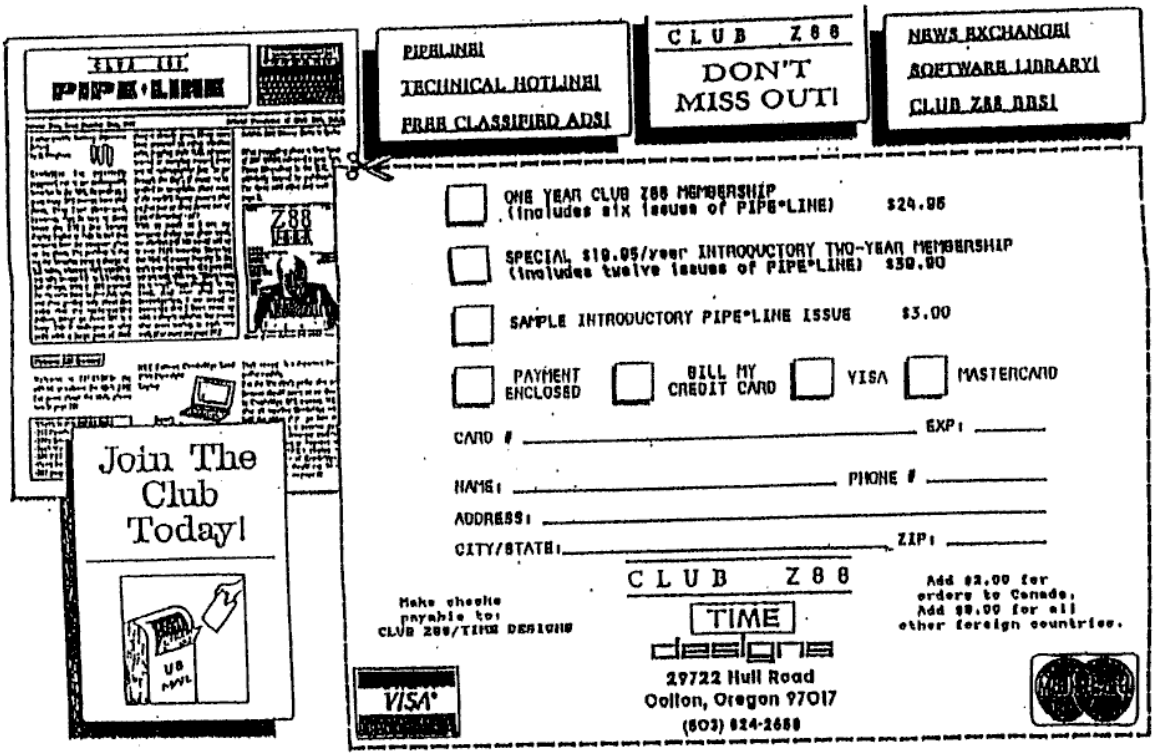
♣ ♠ ♥ ♦ ♣ ♠ ♥

ALSO, THE ONLY PERSON OFFERING PROGRAMS ON CHEAP DISKS!! THEY WILL COST BETWEEN $10 AND $40

♣ ♠ ♥ ♦ ♣ ♠ ♥

THE ONLY PLACE TO GET CAR AC ADAPTERS!

♣ ♠ ♥ ♦ ♣ ♠ ♥



OF SINCLAIR COMPUTERS

ADVERTISING RATE SCHEDULE

effective until July 1989

UPDATE MAGAZINE is published quarterly each January, April, July, and October in support of the users and suppliers of the SINCLAIR QL, SINCLAIR Z88, AND SINCLAIR-TIMEX TS-2068.

The following Ad sizes are based upon a page size of 8 1/2 inches by 11 inches. Ads are black and white only. Reduction and enlarging capability exists for ads that are off size. Camera ready copy should arrive at least 15 days before publishing date.

DISPLAY ADVERTISEMENT RATE SCHEDULE

AD SIZE	PRICE	
2 pages	$220.00	
1 page	120.00	
2/3 page	90.00	(vertical or horozontal)
1/2 page	65.00	(vertical or horozontal)
1/4 page	35.00	(vertical or horozontal)
1/8 page	20.00	(vertical or horozontal)

Special placement of full page or double page ads, add 10% and call for reservation (inside front cover, first page, inside back cover, back cover).

Line ads: 78 characters per line across page, 1 line minimum, $6.00 per line. Will be type set to Elite 12 Chrs per line.

Your existing Ad copy for other publications is fine. Just send copy with instructions for sizing (up, down, or same). Enhancement of legibility is usually possible. Draft quality dot matrix print is not so good.

TERMS: Check Payment with order 10% discount, or billing on publication date. (no credit cards).

Send Ad copy to: UPDATE MAGAZINE, 1317 Stratford Ave., Panama City, FL 32404. (Tel: 904 871 4513)

Update Magazine ISSUE DISK Library

Index by Title and Issue Month of Article Documentation

For the TS-2068

✓ Oct 87/Jan 88 Consolodated Disk
* Mail Merge * List Looker * Purity
* J-Utilities * Extra Memory.Utils

✓ The Mitchell A Disk April/July 88
* Budget * MS-TAS * Other Utilities

✓ Hartung A Disk (presented over 4 issues)
A complete Disk Management group (DOSDEX)
For Oliger Disk Only

✓ DAISY.B6 July 89
A "Whopper" grossenbigenheimer software
that does everything but tie your shoes.

✓ Mitchell VC Disk July 89 PAGE 32
View Calc for Disk Drive

✓ 1989 Update Collection 4 1989 issues
All Utilities published during 1989

✓ D..base-1 July 89
Flexible Data Base

Keep your software library up to date with these Issue Disks and increase your computing power. Software and utilites MAKE a Computer System.

For the Sinclair QL

✓ Cable A-1 Disk Oct 88
* Helper Software * Arithmetic
An outstanding Manager for Disk and Microdrive

✓ Cable A Disk Oct 88
* Refield * Scan
Archive Softwares

✓ Hartung A Disk Oct 88
* Address File program
* Utilities

✓ Cable B Disk July 89
* "Db_easy" Data base software

✓ Hartung B disk Jul 89
* 8 programs * 6 Utilities

✓ Cable B Disk July 89
* "Db_easy" Data base software

✓ Cable C Disk July 89
* Multitasking Manager * Filer program

Issue disks are supplied for the QL on either 5 1/4" floppy or Micro drive cartridge. If ordering for Micro cart, send a formatted cartridge for dubbing. All programs will run in the Un-expanded QL, unless specified otherwise.

Issue disks for the TS-2068 are supplied on 5 1/4" disk in one of two formats, DSDD 40 track or DSDD 80 track. Please specify 40 or 80 track format. Supported are Larken LKDOS, Oliger DOS, and Aerco FD-68 disk systems only.

All issue disks are $22.00 US Funds. Intl Money Order or Checks drawn on US Bank. (postpaid to USA and Canada). Add $5.00 for other Country mailing.

Back Issues of Update Magazine are available in Year Series of 4 issues for $18.00 (postpaid). Two Year Series are available, 1988 and 1989. The 1988 series pertains only to the TS-2068. Update began Sinclair QL coverage with the first 1989 issue. Support of the Cambridge Z88 began in mid 1989 and will begin to be "programming intensive" as the user group develops.

DISK DRIVE SYSTEM SPECIALS

Looking for a disk drive system? LOOK NO FURTHER!

TS-2068

RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OREGON 97045
503/655-7484 * NOON-10 TUE-SAT

TS-2068

SYSTEM #1

LKDOS SINGLE DRIVE UNIT

Here is a GREAT starter setup! If you are just getting into disk drives, this one will allow you to expand later-includes items below:

LARKEN DISK I/F - CABLE
LKDOS CARTRIDGE-DEMO DISK
1-MITSUBISHI 3.5" 400L DD
UTILITY PROGRAMS-FULL DOCS
STEEL CASE W/POWER SUPPLY

Send copy of ad w/ck or MO for

ONLY $240.00 + $8 PH

Order either system this ad and get 5 FREE DISKS! RMG#P102

SYSTEM #2

LKDOS DUAL DRIVE SYSTEM

This system is READY-TO-RUN! Just plug it in and go! Includes ALL of the following items:

LARKEN I/F & DOS CARTRIDGE
2- MITSUBISHI 400K 3.5" DD
STEEL CASE W/ POWER SUPPLY
CABLE & DEMO/UTILITY DISK
FULL LKDOS DOCUMENTATION

All this AND after sale support!
Send copy of this ad w/ ck or MO
You SAVE 10% off regular low price

ONLY $265.00 + $8 PH

RMG#P107

##########

HERE IT IS!

TRACER 2068

A NEW PROGRAMMING UTILITY FOR YOUR 2068!

This little package allows you to SEE WHAT YOUR PROGRAM IS DOING! Imagine being able to watch the program lines go by at the bottom of the screen as the display continues unaffected (other than the speed)!
A GREAT LEARNING TOOL!!
You can see what each command does! What better way to learn the basics of BASIC?

YOU SET THE SPEED! AS SLOW OR AS FAST AS YOU LIKE!
SIMPLE COMMANDS - ONLY 3 OF THEM!

We think that TRACER is the best learning and debugging tool we've seen! We think that you will agree! The working part of the program is 100% MC and it is INTERUPT driven!

Another SUPER product from S & K SOFTWARE, the name that brought you KRUNCHER!®

AT ONLY $14.95+$1.50ph
THIS ONE IS A MUST FOR ALL PROGRAMMERS!
IT WILL SAVE YOU MORE THAN THAT IN TIME ALONE!

TRACER 2068
IS AVAILABLE EXCLUSIVELY FROM

RMG ENTERPRISES

##########

Z88!

The Z88 imports/exports work files from both IBM and Macintosh

We Support the Z88!

We just wanted you to know that we will be supporting the Z88 to the best of our ability! If there is anything that you would like for your Z88 computer, be sure to let us know! As more and more products become available for this fine new portable computer, RMG will have it for you. If we don't, let us know and we will do our best to get it for you—FAST!

For more information or prices, contact us at the address or phone given below:

RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OREGON 97045
503/655-7484 * NOON-10 TUE-SAT

RMG ENTERPRISES

IS PROUD TO ANNOUNCE
A NEW BUSINESS SOFTWARE PACKAGE FOR THE QL!

INVENTORY/INVOICING PACKAGE

FROM EXECUTIVE WORKSHOP

If you are running a small business that depends on a GOOD inventory control program, THIS IS THE ONE FOR YOU! With our new package you can keep track of all SALES, INCOMING STOCK, CURRENT STOCK VALUES-BOTH WHOLESALE AND RETAIL, as well as generate the following hard copies:

INVOICES - BACKORDER MEMOS - PACKING LISTS - SHIPPING LABELS
INVENTORY REPORTS - PURCHASE ORDERS - FLAGGED ITEM REPORTS
PRICELISTS - SELECTIVE PRICE LISTS

All of this and the best part is, WE WILL CUSTOMIZE THIS PROGRAM TO SUIT YOU! When you order this package, you will be sent a short questionaire to fill out and return. Within 2 weeks you will receive your program, ready to go!

ALL THAT AND AT A PRICE OF ONLY **$59.95**+2ph

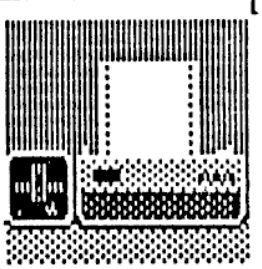
FOR MORE INFORMATION ON THIS AND OTHER FINE QL PRODUCTS CONTACT:

RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OR 97045
503/ 655-7484

(FOR CATALOG - SEND $2 - DEDUCTED FROM FIRST REGULAR ORDER)

NEED A PRINTER?

WE CAN HELP!



RMG can supply many top brands at low prices. Names like: CITIZEN-STAR-ALPS PANASONIC-OKIDATA-EPSON-SEIKOSHA BROTHER AND MORE!

*** CHECK THESE OUT! ***

PANASONIC KX1080i	ONLY $175.00
PANASONIC KX1091i	ONLY $199.95
STAR NX1000 MULTI-FONT	ONLY $180.00
STAR NX1000 RAINBOW	ONLY $239.95
STAR NX2400 24-PIN	ONLY $339.95
SEIKOSHA SP-180AI	ONLY $159.95
CITIZEN 120D	ONLY $159.95
CITIZEN 180D	ONLY $175.00

Please add $15 each for sh. & ins.
We also have MANY used printers in stock-write or call!

RMG ENTERPRISES
1419 1/2 7TH STREET
OREGON CITY, OREGON 97045
503/655-7484 * NOON-10 TUE-SAT

****
SPECIALS! SPECIALS! SPECIALS!
****

WE HAVE SPECIALS!

WE HAVE NEW SPECIALS EACH MONTH!

WOULD YOU LIKE TO BE AMOUNG THE FIRST TO FIND OUT WHAT WE WILL HAVE TO OFFER EACH MONTH?

HERE'S HOW!

JUST SEND US 12 LEGAL SIZED S.A.S.E.s AND YOU WILL BE AMOUNG THE FIRST!

YOU WILL RECEIVE AT LEAST 2 (USUALLY MORE) SHEETS OF SPECIALS, OVERSTOCKS, ONE-OF-A-KINDS, HARD TO FIND ITEMS, ALL NEW PRODUCTS AND ANY SUPER BUYS THAT WE HAVE TO OFFER!

THE BEST PART!!

WE'VE SAVED THIS FOR LAST. JUST IN CASE YOU WERE SAYING TO YOURSELF "WHY SHOULD I PAY FOR THE STAMPS TO GET THESE OFFERS?" WELL LISTEN TO THIS!! YOU WON'T BE PAYING! WE WILL! THAT'S RIGHT!

IF YOU SEND THE ENVELOPES, WE WILL SEND YOU, ALONG WITH THE FIRST MONTHLY MAILING YOU RECEIVE, A COUPON GOOD FOR $5.00 OFF YOUR NEXT ORDER FROM OUR REGULAR CATALOG!

I DON'T SEE HOW WE CAN BE MORE FAIR THAN THAT!

HURRY!

DON'T MISS ANY OF OUR GREAT BUYS!
GET ON THE LIST TODAY!
SEND YOUR ENVELOPES TO THE ADDRESS BELOW!

WE WANT TO SERVE YOUR NEEDS!
IF THERE IS SOMETHING THAT YOU WOULD LIKE TO SEE US CARRY, LET US KNOW! MAYBE WE CAN SAVE YOU MONEY ON IT!!